Iterative K-Line Meshing *via* Non-Linear Least Squares Interpolation of Affectively Decorated Media Repositories

Anestis A. Toptsis* and Alexander Dubitski

Department of Computer Science and Engineering, York University, Toronto, Ontario, M3J 1P3, Canada

Abstract: We present an algorithm that organizes a song repository upon recording a user's memory experiences from previous music listening activities. Our method forms an affectively annotated network of songs. The network's connections correspond to a person's recorded memory experiences related to song preferences when the person is at different states of affective bias. Upon formation of this network, an intelligent affect-sensitive network navigation algorithm synthesizes playlists that conform to desired affective states. The method for the network formation is highly individualized, in the sense that it takes in account an individual's music preferences which are typically subjective and may differ from user to user. Also, the method is content independent, in the sense that it does not rely or favor any particular music genre. In fact, the method is applicable to any type of media, not only songs. We implement our method and present evaluation results from the introspection of our algorithms' execution and from feedback recorded during the evaluation by human test subjects. The evaluation results clearly indicate that the proposed method significantly outperforms the most typical paradigm of random song selection.

1. INTRODUCTION AND BACKGROUND

Organization and retrieval mechanisms for media repositories is a topic of vigorous research activity during the past several years The issues addressed in such research are content-based organization and retrieval mechanisms and efficiency in data access (e.g., [1, 2]). All the reported mechanisms are useful in the sense that they facilitate organization and retrieval based on characteristics of the content of the media, such as the semantic or grammatical relevance of text from a publication, and the relevance of a theme from an image or a video clip. Solutions that facilitate efficient data access are also useful, especially when dealing with large and public media collections.

1.1. Motivation and Related Work

In this paper we cover a different facet of media access. Specifically, we address the issue of media organization and retrieval based on the user's emotional state. The difference between our approach and methods such as the ones in [1, 2], mentioned above is that in our approach the data is organized and retrieved according to its conformity to the user's mood rather than to any structural characteristics of its content. Our method is applicable to any type of media, such as video clips, images, text, and music. For the shake of being specific and evaluating our method, we assume that the type of media is music.

We start with a collection of songs that are available for listening. The storage medium of the collection is not important although, as it is nowadays typical, the songs may be assumed to being stored in a mobile devise such as the iPod or many modern mobile phones. In such an environment it is typical that the user has the option to choose to listen to *random selections* of songs or to select *serial playing*, or to *select individual songs*, one by one as she goes along in the listening process. The random and the serial playing modes are usually the most frequently used since all listening devices provide default settings for both of these modes. However, for devices that contain at least several hundred songs (or more, as it is typical nowadays) a random or serial selection of songs is bound to contain songs that are not as desirable to listen to, based on the listener's current mood. As reported in [3] and also confirmed by our evaluation in section 4, playlists of randomly selected songs have an approximate user satisfaction level of only 20%. The third option-explicit song selection, one song at a time, by the listener, would probably produce an optimal listening sequence. However this option is rather impractical since it requires constant attention and interaction by the user-searching though the collection and selecting the next song to be played. In most real-life cases, a listener is unwilling to continuously interact with the listening devise in searching and selecting "the best next song" and he just lets the devise play whatever comes next. Besides, in many occasions, an interaction between the user and her listening devise is really not possible, such as in cases when the listener is driving, or performing some task that requires her fairly undivided attention.

There are several successful attempts (e.g., [3, 4]) to *automate* the creation of playlists. All these approaches are based on correlating a person's physical activity and music preferences. They all require that a person is actively engaged in some physical activity (such as jogging) during the creation and dispensing of the playlist. The methods are not applicable for cases that the person is at rest, or for types of

*Address correspondence to this author at the Department of Computer Science and Engineering, York University, Toronto, Ontario, M3J 1P3, Canada; Tel: (416) 736-2100, Ext. 66675;
E-mail: anestis.toptsis@gmail.com, anestis@cse.yorku.ca

1874-0618/08

media other than songs. Also, they do not take in account the emotional state of the person.

In this paper we present a method for organizing a collection of songs in such a way that after the collection is organized, the system is able to automatically select, upon request, a series of songs to play, based and conforming to the current emotional state of the listener. Our method forms an affectively annotated network of songs. The network's connections form playlists that correspond to a person's recorded memory experiences with respect to song preferences when the person is at different states of affective bias. The formation of the network of songs is highly individualized, in the sense that it takes in account the individual user's music preferences which are typically subjective and may differ from user to user. Also, the method for organizing the song collection is content independent, in the sense that it does not rely or favor any particular music genre; it can be invariably applied for any kind of music and can be used by any user, subject only to the user's preferences and subjective evaluations of what constitutes "good" or appropriate music. Upon formation of this network, an intelligent affect-sensitive navigation algorithm browses the network and creates playlists that conform to desired affective states.

The remaining of this section provides related background knowledge and the main tools and definitions that we use in our method. Section 2 presents our method of organizing and using a song collection. Section 3 is the evaluation of our method. Section 4 summarizes our findings and discusses future research directions.

1.2. Background

We employ tools found in three related disciplines-Artificial Intelligence (AI), Cognitive Science, and Psychology. From Psychology we use the emotion theory of Ekman [5], which describes universally occurring emotional states of human beings. From AI and Cognitive Science we borrow from the memory theory of Minsky [6, 7] and specifically a structure called K-lines [8]. In recent years, work which combines emotion involving issues and these disciplines has evolved into a new discipline, under the name Affective Computing [9, 10].

The Cognitive Science Aspect

Picard [9, 11] discusses the significance of computers to understand human emotions mainly for the purpose of improving computer-human interactions. One of the possibilities that is pointed in [11] and [9] is to enable a machine to select and play music according to someone's emotions.

The opinions of what is the origin of emotions and how they mesh with intelligence vary widely (examples of rather vastly differing views are [12, 13]), however it is rather commonly agreed that emotions play an important role in intelligence (e.g., [14]). As it was pointed out more than 25 years ago by one of the greatest AI visionaries,

> "*The old distinctions among emotion, reason, and aesthetics are like the earth, air, and fire of an ancient alchemy. We will need much better concepts than these for a working psychic chemistry.*" [15], page 1.

Recently, it is increasingly recognized that emotions are an integral part of many (if not all) intelligence bearing processes such as human-human interaction, decision making, complex task completion, learning, etc. ([9, 10, 16, 17]). As such, contrary to the traditionally held belief (or expressed desire) that intellectual and emotional processes operate in separate and incompatible cognitive realms, it is becoming increasingly accepted that mixing purely intellectual with emotional processing may have benefits which could not be realized by intellectual processing alone. Notable examples that contribute to this belief are Damasio's patient [18, 19], in which a human without emotion processing capability is deemed unable to make even the simplest of decisions and task completion, as well as the decision making and planning scenario outlined in [9] in which it is shown how affective (emotion bearing) heuristics can prune a vast search space dramatically and thus make a decision making and planning task reasonably tractable.

The State of Mind (SM)

There are a number of studies that propose different definitions and classifications of emotions. Among them, Ekman [5] is a widely accepted classification and it is used in several affective computing projects of the highest caliber (e.g., Liu, *et al.* [20], Matsumoto, *et al.* [21]). Ekman [5], Minsky [6], and Picard [9], all argue that this classification consists of *basic* emotions, meaning emotions that are encountered in all humans, regardless of cultural background. In this project, we use the same classification. However, we design our methodology and algorithm in such a way that it is easily amenable to any newer or better classification. The six basic Ekman's [5] emotions are: (1) sadness, (2) happiness, (3) anger, (4) fear, (5) disgust, (6) surprise. Using this classification, we represent a person's emotional state, (or, state of mind (SM)), as a tuple of 6 attributes

SM = < Sadness, Happiness, Anger, Fear, Disgust, Surprise>.

Note, although some of the above emotions seem contradictory, it is not necessarily so. For example, according to [6], "happiness" is not opposite to "sadness". In order to represent different SMs we allow each of the 6 attributes to fluctuate within a range of values. In this project we set this range from 0 to 5. For each emotion attribute, 0 represents the lowest possible intensity of that emotion and 5 represents the highest possible intensity. For example, in SM = < 3, 3, 5, 2, 3, 2 > the left-most number 3 means that the left-most (1^{st}) emotion (sadness) has intensity 3.

Note, with our chosen numbers (6 emotion attributes and 6 integer values per emotion) we allow for 6^6 = 46,656 possible SMs, assuming that we only allow integer values for each emotion attribute. (In our implementation we also allow for non-integer numbers, so the range [0...5] for emotion values effectively allows for more than 6 values.) According to Minsky's theory of memory [6] and [7], the human brain is a collection of a large number of mental agents and each of those agents can be either active or inactive at any particular time point. The geography and arrangement of agent acti-

vation at any given point then represents the SM of that person at that time point. In this sense, our model of SM, with 46,656 possible mental states, resembles Minsky's model of SM. Of course, Minsky's model also deals with the mechanisms that may be used to trigger activation and deactivation of certain mental agents. To the best of our knowledge, there is no reported work that gives a full account of the workings of those mechanisms. There are, however, partial treatments which investigate certain mind activities, based on Minsky's theory. Notable examples of such works are [22-24]. This project does not intend to elaborate or investigate such mechanisms. Similar to Minsky, we have Sloman's mind architecture, e.g. [25, 26]. Although the two architectures seem different on paper, both Minsky and Sloman seemingly agree that their approaches share major common characteristics and differ only in the design representation [27]. We bypass the difficulty of what mechanisms produce what SMs by using Ekman's emotion list (of the 6 basic emotions) and then assuming that any user of our system would be able to express her SM by associating values to each of those emotion attributes, to the best of her understanding. We also note, however, that relying on the user to express her emotional state is not a perfect scenario either. Reportedly [16, 17], the average human does not possess enough self-knowledge to be able to clearly express at any given time what her emotional state is.

The Data Structure

Once we decide how to represent a person's state of mind, we need some structure and mechanism to store and use this SM. For this, we choose a structure called K-lines. The K-lines structure was introduced by Minsky in [8] and it is extensively used as one of the tools that Minsky employs to describe a mind/brain architecture in [6] and [7]. The basic idea is that a K-line is a structure that allows recording and storage of memory experiences that are formed during the process of learning to perform a task. Once a K-line is formed in our brain, future requests for performing the same or a similar task cause our brain to activate an appropriate K-line (provided that such a K-line exists) which consequently allows the person to perform the newly requested task by using the memory experiences recorded in that K-line and thus not having to go through a time consuming (or impractical) learning process again. Quoting from [8],

> *When you "get an idea", or "solve a problem", or have a "memorable experience", you create what we shall call a K-line. This K-line gets connected to those "mental agencies" that were actively involved in the memorable mental event. When that K-line is later "activated", it reactivates some of those mental agencies, creating a "partial mental state resembling the original".*

In its simplest form, a K-line is a chain of K-nodes. Each of the K-nodes represents a memory experience (or a process that has to be followed) that is learned and recorded in the K-node during a person's process of learning to perform a task. As we continuously accumulate memory experiences and knowledge during our lifetime, a vast system of K-lines is formed in our brain. Since many of the experiences of a person are related (or, at least they are not completely different from each other), some of the formed K-lines may contain K-nodes that are identical, within certain degree of analogy or abstraction.

Our K-lines are chains of K-nodes, each of which has the structure shown in Fig. (**1**).

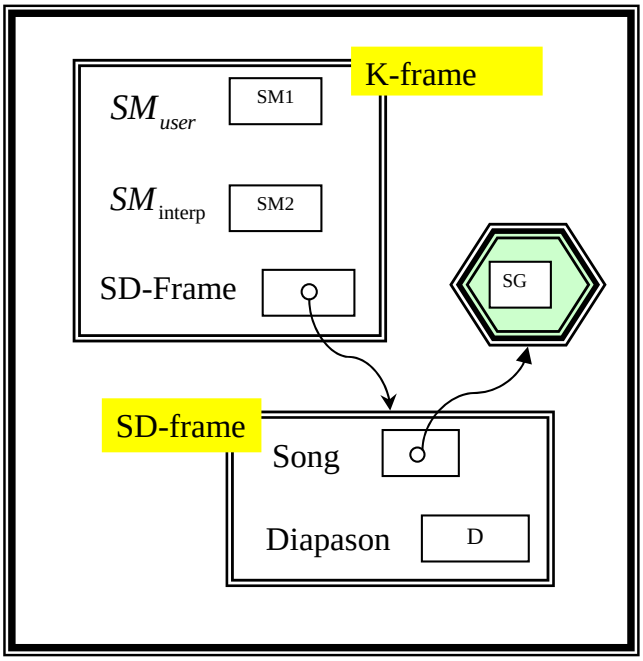

Fig. (1). A K-node.

This structure consists of one main *frame*, K-frame, which contains another frame, the SD-frame. A frame is an AI related structure introduced for knowledge representation, also by Minsky [28]. As described in [28], a frame can contain several placeholders (such as the SM_{user}, SM_{interp}, and SD-frame shown in Fig. (**1**)) and some of those placeholders can be frames themselves (such as the SD-frame placeholder inside the K-frame). In the K-frame, the placeholders SM_{user} and SM_{interp} are SMs expressed as tuples of 6 attributes, according to the earlier discussion. From these two SMs, the SM_{user} is input by the user at some point during our method (described in the next section). The SM_{interp} component is calculated by our algorithm. The *Song* component of the SD-frame is a pointer to a particular song SG from a song collection. The assignment of this pointer is done by our algorithm, as described in the next section. The *Diapason* component of the SD-frame is a special form of SM. Specifically, a diapason has the form

$$D = \langle [v_{1a}, v_{1b}], [v_{2a}, v_{2b}], ..., [v_{6a}, v_{6b}] \rangle$$

where $[v_{ja}, v_{jb}]$ is an interval in which the values v_{ja} and v_{jb} represent acceptable minimum and maximum intensities for the j-th emotion (j=1, …, 6). The number of intervals in

D is 6 since each interval corresponds to one emotion and we use 6 emotions throughout this paper. A song SG with associated (i.e., in the same SD-frame) diapason D is considerd appropriate to be experienced under any SM in which the j-th emotion attribute value falls within the interval $\left[v_{ja}, v_{jb}\right]$. The value of diapason is calculated by our algorithm as described in the next section.

2. PROPOSED METHOD

Our purpose is to form K-lines which consist of K-nodes that have the structure described in the previous section and such that all the K-nodes of each K-line contain songs which are appropriate to be experienced (played or listened to) upon request, when the user is under a specific SM. Our method consists of three parts.

Part A: Data Collection. During this part, a user's listening experiences are recorded into K-nodes and the K-nodes form K-lines. The outcome of part A is a collection of K-lines, as shown in Fig. (**2**). Each of the five K-lines shown in Fig. (**2**) contains several K-nodes. Each K-line has a difference color and boldness for ease of identification. The K-nodes are the blue-shaded circles. Each K-node has the structure described in the previous section and shown in Fig. (**1**).

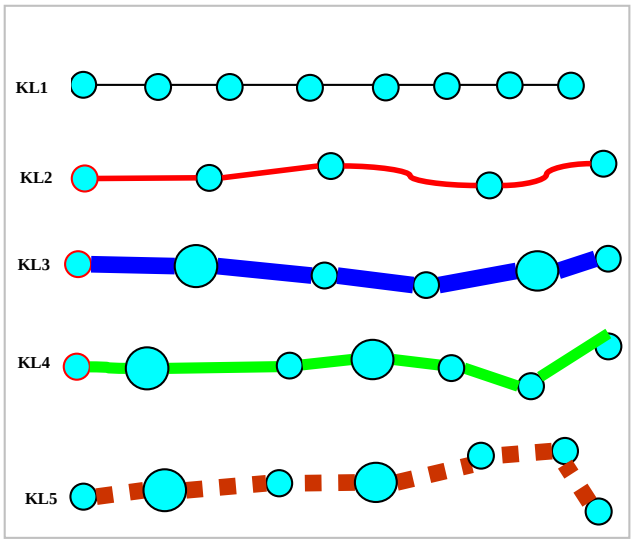

Fig. (2). Five K-lines.

Part B: Formation of K-Line Mesh. This is the heart of our method. During this phase, we process the K-line collection formed in Part A and we calculate and adjust certain parts of the K-nodes. This phase results to the formation of a K-line mesh, similar to the one shown in Fig. (**3**).

Fig. (**3**) shows a K-line mesh formed by the five K-lines of Fig. (**2**).

Part C: K-Line Mesh Navigation. Our algorithm navigates the K-line mesh and assembles a playlist consisting of songs deemed to be a best match (most appropriate to be listened to) with the current SM of the user. This phase is essentially the way that a typical user is using the system and is meant to be performed multiple times, every time that a user wants to listen to songs that comply with her SM at that time. Fig. (**4**) shows a playlist that the navigation algorithm extracts from the K-line mesh of Fig. (**3**).

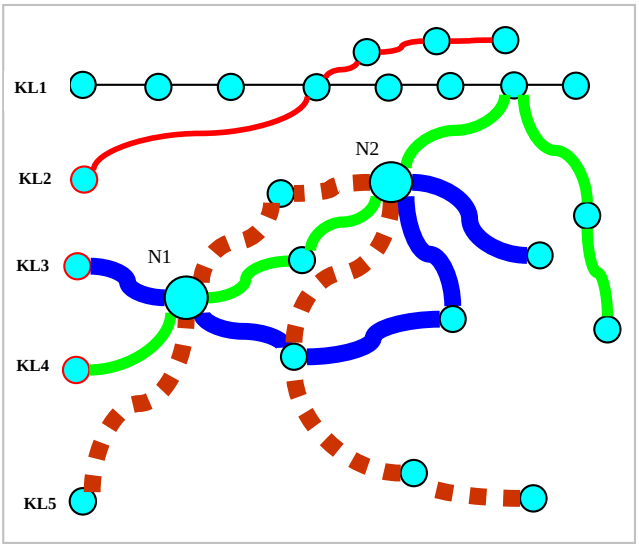

Fig. (3). A K-line mesh.

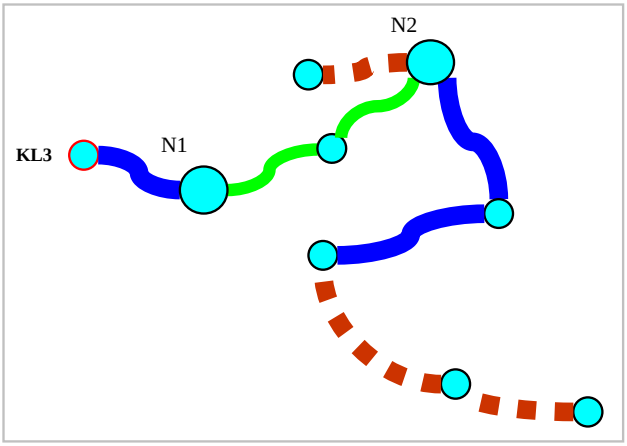

Fig. (4). K-line mesh navigation result.

Next, we describe Part A, Part B, and Part C.

2.1. Part A: Data Collection and Initial Formation of K-Lines

We start with a repository of songs available for listening. The purpose of this phase is to form a K-line collection, such as the one shown in Fig. (**2**). Each K-line of Fig. (**2**) contains several K-nodes. Algorithm A describes how such a collection of K-lines is formed.

Algorithm A

```
1. Periodicity = P;
2. next_Kline_ID = 1;
3. repeat {
       a. form_Kline(next_Kline_ID);
       b. next_Kline_ID++;
4. }until (system or user quits);

(method)
Form_Kline(k) {
1.  count = 0;
```

```
2.  repeat {
3.  user selects a song S;
4.  if (isKlineIntersection(k, S)){
          a. user enters SM_user ;
          b. count = 0;
5.  }
6.  elseif (count % P == 0) {
          a. user enters SM_user ;
          b. count = 0;
7.  }
8.  else { SM_user = null; }
9.  form_Knode( SM_user , null, S, null);
10. count++;
11. } until (system or user quits) ;
12. } // end Form_Kline(k)
```

(method)

Form_Knode(SM_{user} , SM_{interp} , Song, Diapason**)**

- **Input:** the data fields of a K-node per the description of Section 1.
- **Action:** forms a K-node and initializes its fields using the input. Appends the K-node to the K-line that is in progress.
- **Intelligence:** all songs selected during the formation of the same K-line are different.

(method)

isKlineIntersection (k, S)

- **Input:** a K-line ID k and a song S.
- **Returns:** *True*, if song S which is about to be part of a K-node of K-line k was selected earlier during the formation of another K-line. *False*, otherwise.

In the above algorithm, the user selects songs for listening (line 3 in method Form_Kline) and the system prompts the user for her SM, periodically (lines 4.a. and 6.a. of method Form_Kline) and creates a K-node for each chosen song (line 9 in method Form_Kline). The criteria used to prompt the user for her SM are (a) whether or not the currently selected song was also selected earlier during the formation of a previous K-line (line 4 in method Form_Kline) and (b) if a predetermined periodicity frequency P is reached at that K-node (line 6 in method Form_Kline) All K-nodes created by the above algorithm have their SM_{interp} and Diapason fields null (line 9 in method Form_Kline). Some, of the K-nodes have their SM_{user} field initialized (lines 4.a. and 6.a. of method Form_Kline) but SM_{user} is null for the most of the K-nodes (line 8 in method Form_Kline). Also, all K-nodes point to songs that have been selected by the user (line 3 in method Form_Kline). The above algorithm is inspired by a data collection process performed for media organization, described in [29].

At the end of Algorithm A, a collection of K-lines as shown in Fig. (**2**) is formed. Some of these K-lines "intersect", i.e. they contain K-nodes which point to the same song.

Why the Repeated Requests for SM?

One aspect of Algorithm A that may warrant some explanation is why do we repeatedly ask for the current user's SM-either periodically every P K-nodes, or every time that a repeat-song selection occurs (i.e., a K-line intersection occurs). In principle, the formation of a K-line could be done simply by asking for the SM of the listener only at the formation of the 1st K-node of each K-line and then form all the K-nodes of that K-line by just associating with them the selected songs, without ever requesting the user to enter his current SM for the duration of the formation of the same K-line. Since our assumption (and intention) is that each K-line captures one listening experience based on the user's mood which is captured at the 1st K-node of that K-line, it could be both simple and convenient to consider all songs that are selected during the same K-line formation as being appropriate for the SM entered at the 1st K-node of this K-line. Note, such a strategy would also make the formation of K-lines more efficient since there would be no need to periodically interrupt the user (lines 4.a. and 6.a. in method Form_Kline) with requests of SM input during the formation of the same K-line. Unfortunately, such an approach is not very realistic. The reason is that during a song-listening session the SM of the listener, changes multiple times. These changes, as reported in [5], are due to the act of listening to the selected song. Therefore, by the time that the last K-node of that K-line is formed, the SM of the listener would most likely be significantly different from the SM that was reported at the 1st K-node of that K-line. It is therefore unrealistic to believe that all the K-nodes of that K-line contain songs that are compatible to the mood recorded in the head K-node of that K-line. In addition, as mentioned in [9], a *mood* is a *series* of successive emotional states, rather than a single emotional state. In our case, each emotional state is represented by a SM. Hence, if we follow an approach that requests the user's SM only at the 1st K-node and for none of the remaining nodes of each K-line, the memory experiences that would end up being captured by the formed K-lines will be rather meaningless, even though the formation of the K-lines will be very efficient. The ultimate remedy to this problem is to request from the user to enter his current SM at every single song selection-i.e., at the formation of each and every K-node during a K-line formation. Unfortunately, this approach has at least two significant drawbacks. First, it is very inefficient to issue and satisfy a request for an updated SM at every single K-node. Since a typical K-line may consist of 50-100 (or more) K-nodes and our assumption is that a large number of K-lines is formed, it is prohibitively time consuming to request and record a new SM for every single K-node. Second, the act itself of requesting an updated SM so often will misdirect the attention of the user from the listening process and also most likely irritate the listener; the attention loss and the irritation itself will influence the user's mood, resulting to a SM that is contaminated by factors (the loss of attention and the irritation impact) other than the listening experience. Due to the above drawbacks (high cost and undesired side effects due to attention loss and to irritation) we reject the option of requesting a SM during every K-node.

Therefore, having rejected the two extreme options for frequencies of SM requests (only once per K-line and once per K-node) we choose the sole alternative of requesting the SM only periodically, as specified by lines 4.a. and 6.a. of method Form_Kline, in Algorithm A.

2.2. Part B: Formation of K-Line Mesh

At the end of Part A, there are several K-nodes with a SM entered by the user, but there are also many K-nodes without their SM_{user} = null. Also, SM_{interp} = null and Diapason = null for all K-nodes. The purpose of Part B is two-fold. First, we calculate SM_{interp} for every K-node in our K-lines. Second, we calculate the Diapason field for all K-nodes. All steps of Part B are performed without any user interference. At the end of Part B, all K-nodes have all the their fields initialized and the collection of K-lines resembles the one shown in Fig. (**3**). Part B is an iterative process, as shown in Algorithm B.

Algorithm B

repeat {

Step 1: Calculate SM_{interp} for all K-nodes.

Step 2: Calculate *Diapason* for all K-nodes.

Step 3:: Update the interpolated values of step 1

}

until (no significant changes are observed in the Diapason values**);**

Step 1: Calculate SM_{interp} ***for all K-Nodes***

In this step, we calculate the SM_{interp} for all K-nodes of every K-line. Denote by $\langle e_{1jx}, e_{2jx}, ..., e_{6jx} \rangle$ the desired SM_{interp} of the x-th K-node (x=1, 2, …) of the j-th K-line (j=1, 2, …). The sought values in $\langle e_{1jx}, e_{2jx}, ..., e_{6jx} \rangle$ are calculated as

$$e_{mjx} = A + B \cdot M_{mi} + C \cdot M_{mi}^2 \quad (1)$$

where A, B, and C in expression (1) are values that satisfy the following system of equations:

$$A \cdot \Psi^{(w)} + B \cdot \Psi^{(w+1)} + C \cdot \Psi^{(w+2)} = \sum_{i=1}^{n} M_{mi} \cdot i^w \quad (2)$$

where, $\Psi^{(k)} = \sum_{i=1}^{n} i^k$,

for w = 0, 1, 2.

In the above system of equations, M_{mi} is the value of the m-th emotion attribute ($m \in \{1,...,6\}$) of the SM_{user} of the i-th K-node (i starting from 1) that is populated with a SM_{user} within the j-th K-line. Note, the number of K-nodes with SM_{user} is less than the total number of K-nodes in the j-th K-line. *n* is the number of K-nodes whose SM_{user} is not null within this K-line. Note, n is less than or equal (in most cases, less) than the total number of K-nodes of this K-line, since not all K-nodes have a user assigned SM at the end of Part A. The system of equations shown in expression (2) is produced by applying the Least Squares Method (LSM) on each of the K-lines resulting from Part A. The LSM is a type of regression analysis introduced by Gauss around 1794 [31]. Descriptions of the method can be found in various textbooks, including [32-34]. We now give the details of how this method is applied for the calculation of the values for each SM_{interp}.

Recall, during the initial formation of K-lines in Part A, the user is prompted periodically (but not always) to enter his SM as each K-line is created. As a result, after the K-lines are formed at the end of Part B, many of the K-nodes do not have any SM associated with them. Fig. (**5**) shows an example K-line at the end of Part A. the numbers 1, 2, 3, …, 7 denote the K-node index (and ID) within the K-line. The shown SMs are all SM_{user}, entered by the user during Part A. Observe that only K-nodes 1, 5, 6, and 7 have their SM_{user} initialized; and none of the nodes has SM_{interp} (i.e., all SM_{interp} are null).

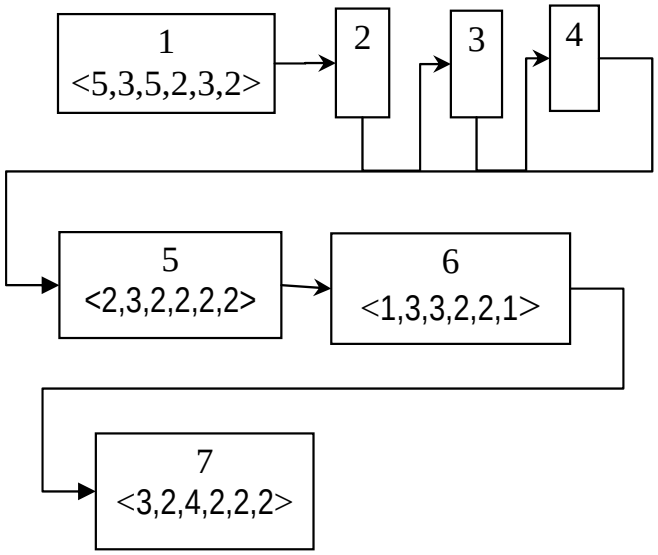

Fig. (5). A K-line.

Fig. (**6**) shows a diagram derived from Fig. (**5**). This diagram is used to explain the interpolation process, i.e., the K-Line interpolation done with LSM using the quadratic equation shown in expression (1).

The horizontal axis of Fig. (**6**) contains integer values which indicate the position (index) of each K-node within this K-line. The vertical axis of Fig. (**6**) contains values representing the intensity of an associated emotion value in the SM_{user} of the K-node matched in the horizontal axis. Fig. (**6**) is derived from the K-line shown in Fig. (**5**) and corresponds to the 1st emotion of each of the SMs that appear in the K-nodes of Fig. (**5**). For example, the left-most plotted point in Fig. (**6**) is the point [x=1, y=5] because the 1st K-node in Fig. (**5**) (i.e., the K-node with index x=1 in Fig. (**6**))

has a SM in which the value of the 1st emotion is 5 (i.e., y = 5). Using the notation introduced in (2), the left-most plotted point in Fig. (**6**) is the point [1, M_{11}]; the 2nd plotted point is the point [6, M_{12}] because the **2**nd K-node with its SM_{user} non-null is K-node 5; and so on.

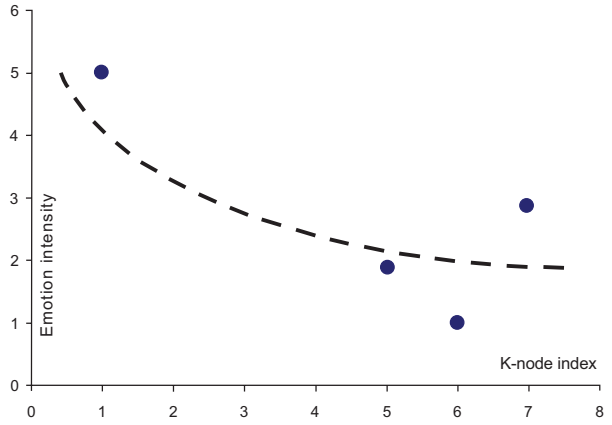

Fig. (6). Plotted points of the 1st emotion attribute of the K-line of Fig. (**5**) and LSM-estimated curve.

Note, not all K-nodes in Fig. (**5**) have SMs associated with them (specifically, K-nodes 2, 3 and 4 do not have SMs). The purpose of this step (step 1) is to calculate interpolated values for all K-nodes 1, …, 7. This is done by calculating a "best-fit" curve as shown by the dashed line in Fig. (**6**), that has an overall minimum distance from the existing plotted points. Applying the LSM mentioned above. The function that we use to calculate such a curve is

$$F\left(x\right) = A + Bx + Cx^2 \tag{3}$$

where A, B and C are unknown values that need to be determined. Once the values A, B, C are found, (see below), then each of the points [i, F(i)], for i=1, …, 6 represents an estimate of the intensity of the emotion attribute under consideration, of the i-th K-node. In particular, for i = 2, 3, and 4, the points [i, F(i)] are estimates for the emotion intensities for which the plotted points are missing in Fig. (**6**). Note, Fig. (**6**) corresponds to the 1st emotion of each of the SMs contained in the K-nodes of Fig. (**5**). As part of this step (step 1) we will have to construct, similar to Fig. (**6**), five more figures, one for each of the other emotions of each SM of Fig. (**5**), i.e., one figure for the 2nd emotion, one figure for the 3rd emotion, etc; and for each such figure, we will have to perform the LSM calculation so that we calculate the "best fitting" curve and thus fill in any of the missing plotted points. Once the best fitting curves are calculated for one K-line, the process is repeated for each of the remaining K-lines. That is, if there are 10 lines formed, then the interpolation process will be performed 60 times and 60 curves will be calculated. A curve shown in expression (3) has minimum distance from the plotted points of Fig. (**6**).

We describe the process for one of the K-lines-the rest being the same. Using the notation introduced in (2), M_{1i} is the value of the 1st emotion of the i-th K-node whose SM_{user} is non-null. The curve of expression (3) has minimum distance from the plotted ponts of Fig. (**6**) when the expression

$$E = \sum_{i=1}^{n}\left[M_{mi} - F(i)\right]^2$$

$$= \sum_{i=1}^{n}\left[M_{mi} - (A + Bi + Ci^2)\right]^2$$

is minimum. In other words, the sum of the squares of the distances (with respect to the horizontal axis) between the curve and the points $P_i : \left(x, M_{mi}\right)$, is minimum (hence the name Least Squares Method). Applying standard calculus, *E* attains minimum values when its derivatives with respect to A, B, C are zero, i.e.,

$$\frac{\partial E}{\partial A} = \frac{\partial E}{\partial B} = \frac{\partial E}{\partial C} = 0\,. \tag{4}$$

Equations (4) yield the system (2) of three linear equations shown earlier. Solving this system with respect to A, B, C, yields the unknowns A, B, C and thus we have the expression for the curve of expression (3).

Step 2: Diapason Calculation

At the end of step 1, every K-node in our K-lines has associated with it either one SM (the SM_{interp}, calculated by the interpolation process) or two SMs (the SM_{interp} calculated by the interpolation process and the SM_{user} as originally entered by the user). The purpose of this step is to create a *range* of appropriate emotion values for each K-node, instead of just having a single value indicating the appropriateness of any song associated with each K-node through its SM, or just having two separate SMs with concrete values, both associated with a K-node. The resulting range of such values is the Diapason of the corresponding K-node. Since, in general, more than one K-node can point to the same song, the Diapasons of all K-nodes that point to the same song are made equal.

Note, the meaning of a SM associated the a song SG is that SG is appropriate for listening when a listener is at that SM. specifically, in cases that a K-node has both its SM_{user} and SM_{interp} intitialized at the end of step 1, the song pointed to by that K-node is deemed appropriate to be listened to if the listener is at SM= SM_{user}; regarding the SM= SM_{interp}, the song is *probably* (but not for sure) appropriate to be listened to if the listener is at SM= SM_{interp}, since the SM_{interp} is an estimated, rather than a user entered value. The presence of two SMs inititialized in a K-node is an advantage, in the sense that it gives two (instead of only one) alternative SMs signaling the appropriateness of that song. At the same time, it raises the following issue: what happens if the listener is at a SM which is *close*, but *not exactly* equal to either SM_{user} or SM_{interp}? The same issue becomes more profound for those K-nodes that have only one SM, the SM_{interp}, initialized at the end of step 1. Specifically, if the listener's SM happens to be

close, but *not exactly* equal to that one SM, is the corresponding song considered appropriate to be listened to or should it be outright rejected? The purpose of this step (step 2) is to address the above issues. Specifically, we create the Diapason for each song in our K-lines collection. The Diapason is a special form of SM as shown at the end of section 1. Even though the Diapason is one of the fields of *every* K-node in the K-line collection, it is considered to be associated with the song component of that K-node. That is, all Diapasons of all K-nodes that point to the song are equal by the end of step 2 (and Part B). Due to this characteristic of the Diapason, if a song is selected, based on its Diapason, later on by our retrieval mechanism, there will be no controversy regarding which K-node among the ones that point to that song contains the most appropriate Diapason for the selected song. For every song SG in the K-line collection, denote

$$T(SG) = \left\{ KN_1, KN_2, ..., KN_j \right\}$$

the set of all K-nodes pointing to SG. The size, *j*, of the set T(SG) is the number of K-lines that interesect at any of those K-nodes. Note, j=1 for those songs that are pointed to by K-nodes whose K-lines do not intersect with other K-lines. Denote by a_{xi} the value of the i-th emotion attribute (recall, the 1st emotion attribute is "sadness") of the SM_{user} of K-node KN_x, and by b_{xi} the value of the i-th emotion attribute of the SM_{interp} of K-node KN_x, for a given set T(SG). For each song SG in the K-line collection, define

- $a = \min_j \left\{ a_{ji} \right\}$, the min a_{xi}, across all K-nodes KN_x, x=1, …, j, of the set T(SG) for song SG.
- $b = \max_j \left\{ a_{ji} \right\}$, the max a_{xi}, across all K-nodes KN_x, x=1, …, j, of the set T(SG) for song SG.
- $x = \min_j \left\{ b_{ji} \right\}$, the min b_{xi} across all K-nodes KN_x, x=1, …, j, of the set T(SG) for song SG.
- $y = \max_j \left\{ b_{ji} \right\}$, the max b_{xi} across all K-nodes KN_x, x=1, …, j, of the set T(SG) for song SG.

Define

$$W(SG,i) = \left[\min\left\{ a, x \right\}, \max\left\{ b, y \right\} \right].$$

W(SG,i) is an interval of values that represent intensities of the i-th emotion attribute for a given song SG. The left boundary of this interval is the minimum intensity of the i-th emotion attribute *ever* reported by the user or calculated by the interpolation process during PartB.step1 for song SG The right boundary of W(SG,i) is the maximum such value. ***Special case***: At the end of PartB.step1, not all K-nodes have their SM_{user} initialized. Specifically, this happens for any K-node $KN_{special}$ that points to a song SG such that T(SG) = { $KN_{special}$ } *and* $KN_{special}$ is not at the periodicity boundary P of Algorithm A. In such cases, the size of T(SG) = 1, i.e., $KN_{special}$ is not at the intersection of any K-lines. Consequently, the system does not prompt the user for her SM_{user} during the creation of $KN_{special}$ in Algorithm A. Therefore, only the SM_{interp} is initialized, during Algorithm B, for K-node $KN_{special}$. Moreover, $KN_{special}$ has x=y since there is *only one* SM_{interp} for $KN_{special}$. This means that the interval W(SG, i) for song SG pointed to by K-node $KN_{special}$, becomes W(SG, i) = [x, x], i.e., collapses to a single value, x.

Note, W(SG,i) fits the description of the i-th interval $\left[v_{ia}, v_{ib} \right]$ of the Diapason D as given at the end of Section 1. In principle, step 2 can stop here and W(SG,i) can be defined to be the Diapason sought in this step. However, upon closer observation, such a final definition of D has some undesirable features. The first undesirable feature is for cases that *x < a and y > b*. In these cases W(SG,i) is equal to [x, y], and [x, y] *contains* the entire [a, b]. Such a W(SG,i) is *unnecessarily too wide*. Note, [a, b] represents a *user specified* interval of emotion intensities deemed appropriate for song SG. [x,y] represents a *calculated* (by the interpolation process) such interval. We argue that the user's wishes (values a and b) have precedence over the estimated values (x and y) derived from the interpolation process. Therefore, in this case, W(SG,i) = [x,y] is considered to be unnecessarily too wide and its size should be reduced so that it becomes closer to the size of [a,b]. The second undesirable feature is for cases when |a-b| is "very small" *and* a < x *and* b > y. In these cases, W(SG,i) = [a,b] *and* W(SG,i) is *too narrow*. In fact, if the K-line of K-node KN pointing to SG happens *not* to have any intersecting K-lines at node KN, a = b and |a-b| = 0. In such cases the interval [a,b] collapses to a single value. Similar to this is the *special case* identified above, for K-nodes such as node $KN_{special}$, for which x = y and |x-y| = 0, and thus the interval [x, y] collapses to a single value.

To remedy the above concerns, we define a *Default Diapason*. The Default Diapason, DD, is a number that represents the size of a hypothetical ideal Diapason and a guideline which all W(SG,i) intervals strive to achieve. Specifically, if the size of W(SG,i) is bigger than DD, we shrink W(SG,i) so that its resulting size is as close as possible to DD; and if the size of W(SG,i) is smaller than DD, we expand W(SG,i) so that its resulting size is equal to DD. Note, when we shrink W(SG,i), it is not always desirable to shrink it enough for its size to reach DD. This happens in cases when the size of [a,b] is *already* bigger than DD. In these cases, further shrinkage of W(SG,i) will result to an interval which does not include a, or b, or either. We consider the importance of values a and b to have precedence over the importance of the DD value for the same reasons (mentioned above) that we consider a and b more important than x and y.

Based on the above discussion and definitions, Algorithm "Diapason calculation", next, specifies the calculation of the Diapason for all songs in the K-line collection.

Algorithm "Diapason Calculation"

```
Set DefaultDiapason DD;
// D is the calculated Diapason
for (each song SG) {
   1. T = T_set(SG);
   2. for (i = 1 to 6) {
         a. W_i = W_interval(SG,i);
         b. W_i_size = the size of W_i;
         c. d = |DD-W_i_size|;
         d. if (W_i == DD)
               i. D_i = W_i;
         e. if (W_i < DD)
               i. D= expand(W_i, d);
         f. if (W_i > DD)
               i. D_i=shrink(W_i, d);
   3. } //end for (i = 1 to 6)
   4. D = <D_1, D_2, …, D_6 >;
} // end for (each song SG)
```

(Method)

T_set(SG)

- **Input**: a song SG
- **Returns**: the set T(SG) of K-nodes pointing to SG, as described above.

(method)

W_interval(SG, i)

- **Input**: a song SG and a
- **Returns**: the set T(SG) of K-nodes pointing to SG, as described above.

(Method)

expand(W, d)

- **Input**: W is an interval of type W(SG,i) as described above; d is a number indicating the amount of expansion.
- **Returns**: the expanded interval W.
- **Action**: this method increases the size of W by expanding it from the left and from the right by a *total* amount d. The size of the resulting interval is equal to the size of the default diapason DD. The expansion is done equally from both sides of W. Note, an expansion occasionally causes one of the boundaries of W to become either less than 0 (which is the minimum possible value for any emotion attribute in our system) or greater than 5 (the max possible value for any emotion attribute in our system), but not both. In such cases, the interval maintains its acquired width, but it is adjusted by sliding it to the left or to the right, as necessary, so that none of its boundaries is less than 0 or greater than 5. It is easy to show that such sliding never causes a violation of the opposite boundary.

(Method)

shrink(W, d)

- **Input**: W is an interval of type W(SG,i) as described above; d is a number indicating the amount of shrinking.
- **Returns**: the shrunk interval W.
- **Knowledge**: this method is aware of the quantities a, b, x, and y that are related to interval W, as described above.
- **Action**: this method reduces the size of W by shrinking it from the left and/or from the right by an amount *not to exceed* d. The size of the resulting interval is no less than the size of the default diapason DD. Also, *a* and *b* (the user-entered values for emotion intensities) are never left outside the resulting interval. The details of how the shrinking of W is performed depend on the relative order of a, b, x, and y, and are as follows. Define z1 = |y-b| and z2 = |a-x|.
  - If the ordering is "**x y a b**", then shrink W from the left only, by an amount equal to min {d, z2}.
  - If the ordering is "**a x b y**", then shrink W from the right only by an amount equal to min {d, z1}.
  - If the ordering is "**x a y b**", then shrink from the left only by min {d, z2}.
  - If the ordering is "**x a b y**", then shrink from the left and from the right, for a total amount of min {d, z1 + z2}.
  - If the ordering is "**a b x y**", then shrink from the right only, by min {d, z1}.

At the end of Algorithm "Diapason Calculation" the Diapason D of each song has the form

$$D = \langle D_1, D_2, ..., D_6 \rangle$$

(line 4 of Algorithm "Diapason Calculation"), where each D_i has the form $[v_{ia}, v_{ib}]$, as described at the end of Section 1; and all K-nodes that point to the same song have the same Diapason D.

Step 3: Update of the Interpolated Values of Step 1

In step 3 we adjust, as necessary, the interpolated values calculated in step 1. Assume

$SM_{\text{interp}} = \langle a_1, a_2, ..., a_6 \rangle$ and

$D = \langle D_1, D_2, ..., D_6 \rangle$, $D_i = [v_{ia}, v_{ib}]$

are the SM_{interp} calculated by step 1 and the Diapason calculated by step 2, for a K-node. In this step we set

$a_i = v_{ia}$, if $a_i < v_{ia}$, and $a_i = v_{ib}$, if $a_i > v_{ib}$. In other words, for every K-node, we adjust the interpolated values of that K-node to fall inside the corresponding Diapason interval of that K-node.

The motivation for doing step 3 is to improve on the estimations of the values done by the interpolation process of step 1. Recall, the interpolation process calculates a best-fit curve based on existing plotted points representing user entered values (Fig. **6**, Section 2). The quality of the SM_{interp} depends entirely on the quality of that curve, i.e., on the extend to which the calculated best-fit curve is an accurate prediction of the true SM of the user at the corresponding K-node. On the other hand, the Diapason associated with a song SG, as calculated in step 2, captures the *collective knowledge of all K-nodes* of the set T(SG). Therefore, if a LSM-calculated value is outside the corresponding Diapason range, this is evidence that the LSM-calculated value is possibly off and, thus, it is adjusted.

Note, the update of the interpolated values in step 3 produces new interpolated values. These new values are the values that are used for the next iteration, in step 1.

Steps step 1, step 2 and step 3 are repeated until two successive executions of the <step 1; step 2; step 3> sequence produce negligible changes in the diapasons throughout the collection. When this happens, the system is considered to be trained and the K-line mesh to be fully formed. Fig. (**7**) shows such a K-line mesh, containing 5 K-lines KL1, …, KL5.

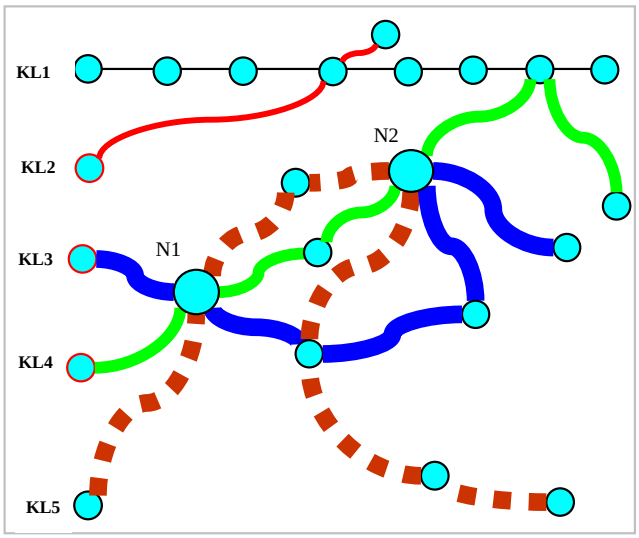

Fig. (7). A K-line mesh.

The nodes shown to be at the intersection of K-lines in Fig. (**7**) (e.g., nodes N1 and N2) are K-nodes that point to the same song. For example, for node N1 which is at the interesection of K-lines KL3, KL4, and KL5, it means that there is a song SG which is selected by the user during the formation of these 3 K-lines and (the subfrane SD-frame of) node N1 points to that song SG. Note, during the K-line formation process, each of those 3 K-lines maintains a separate copy of node N1. The song component of the SD-frame of each such copy of N1 in the 3 separate K-lines points to the same song SG, but the Diapason component of each SD-frame holds a different value. During the iterative process of the repetitive execution of steps 1, 2, and 3 in Part B, those Diapason values change and approach each other during each iteration, until at some iteration they are close enough (within a system defined threshold) to be considered equal to a common Diapason value D. Once this convergence occurs, the three copies of node N1 can be represented by a single node N1 whose SD-frame points to the common song SG and has a Diapason value D. Fig. (**8**) shows how this perceived common SD-frame of node N1 looks like.

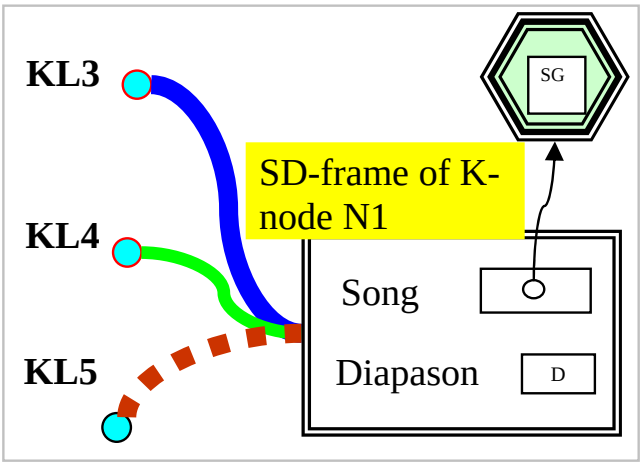

Fig. (8). K-line intersection.

Note, Fig. (**8**) resembles the part of Fig. (**7**) that shows node N1 at the intersection of K-lines KL3, KL4, and KL5.

2.3. Part C: K-line Mesh Navigation

We now describe the process of mining songs from the formed K-line mesh. The process is divided into 4 steps.

Step 1: The user enters a SM, SM_{input}, which captures her present mood and according to which the system should retrieve and play songs. For example, SM_{input} = <3, 1, 4, 3, 5, 2>.

Step 2: The system chooses one of the K-lines that is a *closest match* to the input SM. to determine the closest matching K-line the system calculates the distance between SM_{input} and each of the 1st K-nodes of the existing K-lines and chooses the K-line produces the shortest distance. The distance between SM_{input} and the *1st K-node* of a K-line *KL* is calculated as

$$dist\left(KL, SM_{input}\right) = \sqrt{\sum_{i=1}^{6}\left(\frac{u_i + v_i}{2} - w_i\right)^2} \qquad (1)$$

where u_i and v_i is the i-th emotion attribute value of the SM_{user} and SM_{interp} of the *1st K-node* of K-line KL and w_i is the i-th emotion attribute value of SM_{input}. That is,

$$SM_{user} = \langle u_1, \ldots, u_6 \rangle,$$
$$SM_{\text{interp}} = \langle v_1, \ldots, v_6 \rangle,$$
$$SM_{input} = \langle w_1, \ldots, w_6 \rangle$$

Based on the distance calculation of expression (1), the closest matching K-line is the one whose 1st K-node produced the smallest distance among all the 1st K-nodes of all K-lines.

Step 3: Once the best matching K-line is determined, the system starts from the 1st K-node of that K-line and plays the song pointed to by that node. Then it traverses that K-line sequentially and plays only those songs that are pointed to by K-nodes for which at least L out of the possible 6 Diapason intervals (L=6, initially) cover the corresponding emotion values of $SM_{input} = \langle w_1, ..., w_6 \rangle$. These are the songs which have a Diapason

$$D = \langle [v_{1a}, v_{1b}], [v_{2a}, v_{2b}], ..., [v_{6a}, v_{6b}] \rangle$$

such that the condition

$$w_i \in [v_{ia}, v_{ib}], i \in \{1, \cdots, 6\}$$

is satisfied for at least L out of the possible 6 values of *i*. The number L is the Matching Level in this algorithm. The highest matching level is 6. The minimum Matching Level that we tolerate is 3 (see Step 4, below).

Step 4: This step is performed as part of step 3. As the iterator moves from the current K-Node to the next K-node during step 3, the system checks if the currently visited K-node is at the intersection of two or more K-lines. If this is so, then the system decides to which K-line among the intersecting K-lines to move next. This decision is made similar to step 2 above. That is the system calculates the distance between the SM_{input} and each of the K-nodes that are candidates to be selected next. Then, it resumes by following the K-line that contains the K-node that produces the minimum distance. In case that the entire K-line mesh has been traversed and the created playlist does not contain enough songs, then decrement the Matching Level L by one and repeat step 3, unless the decremenet causes the Matching Level to fall below 3. Algorithm C, next, summarizes the above steps.

Algorithm C

```
/**
Note, method ClosestKline is overloaded. There is one version of this method with one parameter and one version with two parameters. The two versions have different functionality, as described under each of the corresponding methods, below.
**/
Input SM_input;
KL = ClosestKline(SM_input);
CurrentKLine = KL;
MatchingLevel = 6;
CurrentKNode = 1st K-node of CurrentKLine;
Repeat {
   Repeat {
      If (CurrentKLine has no more K-nodes)
         KL = ClosestKline (SM_input) ;
      Else {
         Condition1 = Matches(SM_input, CurrentKNode, MatchingLevel);
         Condition2 = song SG pointed to by CurrentKNode is not in Playlist;
         If (Condition1 ==True and Condition2 ==True)
            Playlist = Playlist + {SG};
         If (isKLineIntersection (CurrentKNode)
            CurrentKLine=ClosestKLine(CurrentKNode, SM_input);
            CurrentKNode = next K-node of CurrentKLine;
      }     // end else
   }   until (playlist size is satisfactory or entire K-line mesh is traversed);
   MatchingLevel--;
   If (MatchingLevel < 3) exit;
} until (playlist size is satisfactory) ;
```

(Method)
ClosestKLine (SM_input)

- **Input**: A SM, SM_input.
- **Returns**: A K-line that is the closest match to the SM_input, as described in step 1.
- **Intelligence**: upon several invocations this method returns a best matching K-line which has not been selected in any previous invocation.

(Method)
ClosestKLine (SM_input, KNode)

- **Input**: A SM, SM_input; a K-node, Knode. **Condition**: Knode is at the intersection of several K-lines.
- **Returns**: A K-line which is the closest match to the SM_input, starting from the Knode, as described in step 4.
- **Intelligence**: upon several invocations this method returns a best matching K-line which has not been selected in any previous invocation.

(Method)
Matches (SM_input, Knode, ML)

- **Input**: A SM, SM_input; a K-node, Knode; the Matching Level ML for this invocation, as described in step 3 and step 4.
- **Returns**: *True*, if the SM_input falls within at least ML out of the 6 intervals of the Diapason of the Knode, as described in step 3; *False*, otherwise.

(Method)
isKLineIntersection (Knode): (Method)

- **Input**: A K-node, KNode.
- **Returns**: *True*, if the input Knode is at the intersection of two or more K-lines in the K-line mesh; *False*, otherwise.

3. EVALUATION

The algorithms and method presented in Section 2 (Part A, Part B, and Part C) are implemented (in Java) on a typical notebook computer. Our evaluation covers three different dimensions: (1) behaviour of the algorithm of Part B for the formation of the K-line mesh. (2) Quality of the K-line mesh navigation algorithm of Part C, in terms of its ability to deliver playlists that conform to a desired SM. (3) comparison of the overall quality of our method against the most frequently used typical alternative of random song selections.

The evaluation of the K-line mesh formation and retrieval of songs is done in two stages, E-Stage I and E- Stage II. E-Stage I corresponds to Part A together with Part B of section 2-i.e., the initial creation of the K-lines followed by the creation of the K-line mesh. In E-Stage II the quality of the created K-line mesh is judged. As it is customary in projects of this type (e.g., [3, 4, 35]) human volunteers are used as test subjects. Eight volunteers are used in our evaluation. Each volunteer performs both E-Stage I and E-Stage II. That is, each of the participants trains the system by selecting songs that the participant judges to be fitting to her SM (E-Stage I), and then the participant judges if the song selections offered by the system are indeed appropriate to her SM (E-Stage II). The reason that the same participant performs both E-Stage I and E-Stage II (instead if using different participants for E-Stage I than the ones we used for E-Stage II) is that music tastes and preferences may vary widely among different individuals and also opinions regarding what kind of music is more fitting for certain SMs may also be a highly subjective matter. As such, it would be rather meaningless to form a K-line mesh based on one person's music preferences and SMs (in E-Stage I) and then to use a different individual to evaluate the appropriateness of song selections offered by the system (during E-Stage II).

Songs Collection

For evaluation we use a collection of 300 songs. Care is taken, after interviewing the users, so that the collection contains only types of songs that all the participants would like to listen to. Since our system is designed to organize *private* song collections (i.e., song collections of individual users, as opposed to song collections designed for public use) it is realistic to assume that a user includes in her song collection songs (or type of music) that she likes to listen, at least from time to time. According to [30] the song's length has a very low impact on the six Ekman emotions. As such, we trim the length of every song to exactly 30 seconds.

E-Stage I: Formation of K-Line Mesh

In this stage the user trains the system by listening to songs and entering her SM periodically, as described in Part A of section 2, using an interface shown in Fig. (**9**).

Each user performs 6 sessions, and each session corresponds to one K-Line formed by the system. Each K-Line consists of 10 K-Nodes and exactly 10 songs. Songs are not repeated during the same session, i.e., all songs within the same K-line are different. At the end of E-Stage I, a mesh of 6 K-Lines is formed with 10 K-nodes in each K-line. The

Fig. (9). User interface for song selection.

number of songs in the K-line mesh is *at most* 60, since repeat-songs may occur across different K-lines due to K-line intersections. As the user selects songs, Part A of our algorithm (formation of K-lines) is executed. Once the user completes the song selection process, Part B of our algorithm is executed and forms the K-line mesh. Recall, Part B includes the iterative process which is the repetition of <step1; step2; step 3>. At this point of E-Stage I we monitor the performance of the iterative process. In all our tests we find that the differences in the calculated Diapasons between successive iterations become negligible after at most 5 iterations. Figs. (**10**) and (**11**) show the impact of successive iterations on the Diapason values.

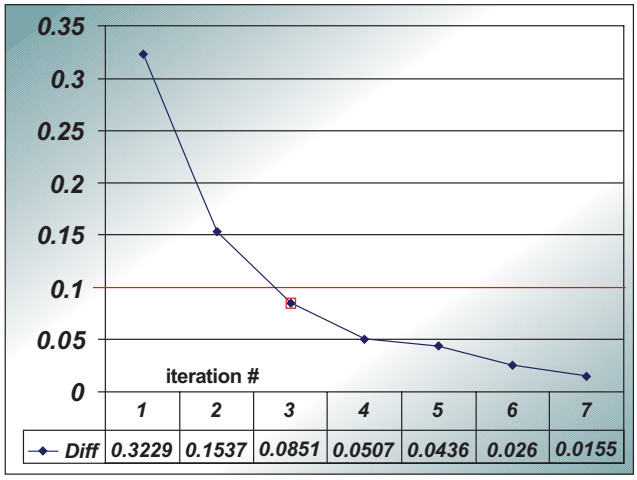

Fig. (10). δ_{ik} fluctuation.

Each plotted point of the curve shown in Fig. (**10**) is the point

$$p_k = \left(k, \max_i \left\{\delta_{ik}\right\}\right), \text{ k = iteration \#}$$

where $\delta_{ik} = \left|D_i^{(k-1)} - D_i^{(k)}\right|$, and with $D_i^{(j)}$ being the i-th interval of a Diapason at the end of iteration j. Recall, the Diapason of every K-node consists of 6 such intervals. During each iteration k of algorithm Part B, our K-line collection contains 360 such intervals, $D_1^{(k)}$, $D_2^{(k)}$, …, $D_{360}^{(k)}$ (since our collection contains 360 K-nodes). To plot the point p_k (k in {1, 2, …, 7}) in Fig. (**10**), we compute $\delta_{ik} = \left|D_i^{(k-1)} - D_i^{(k)}\right|$, for i=1, …, 360. (The difference D1-D2 between any two Diapason intervals D1 = [b1, b2] and D2 = [b3, b4] is computed as max {|b1-b3|, |b2-b4|}.) The plotted point p_k represents the maximum difference observed in a Diapason interval, across the entire K-line collection, as the interval is adjusted from iteration k-1 to iteration k, during the execution of Algorithm Part B. As we see in Fig. (**10**), the maximum difference decreases during successive iterations. This means that during the execution of Algorithm Part B *the Diapasons stabilize across the entire K-line collection*. (Fig. (**10**) displays 7 iterations for illustration purposes. Actually, the algorithm stops as soon as the maximum difference drops below 0.1. This happens at the 3rd iteration in Fig. (**10**), i.e., just below the red line. Note, 0.1 is a realistic threshold, i.e., a small enough difference, since it is only 2% of the entire width of the range of values ([0...5]) covered by any emotion attribute).

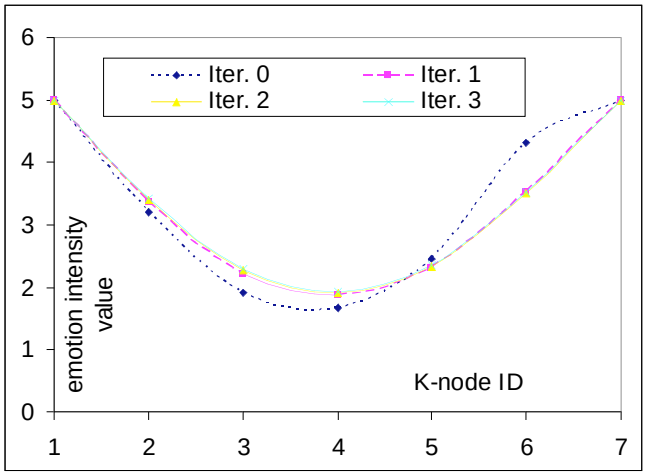

Fig. (11). SM_{interp} fluctuation. K-line 4, emotion attribute 4 ("fear").

Fig. (**11**) shows the changes in the interpolated values (i.e., values of emotion attributes from the SM_{interp} field of each K-node) during the course of several iterations of Algorithm Part B. Recall, in step 3 of Algorithm Part B, the interpolated values of each K-node are updated based on the Diapason computed during step 2, and then the new interpolated values are used for the next iteration. Each curve in Fig. (**11**) corresponds to a single iteration. As we see, the curve for iteration 0 (represented by the blue dashed line in Fig. (**11**)) is the only curve which is clearly distinguishable. The curves for iterations 1, 2, and 3 overlap almost completely. In fact, the curves for iterations beyond iteration 3 overlap even more so and are not displayed in order to avoid cluttering Fig. (**11**). This means that the values of the SM_{interp} during step 3 become closer with the pass of every iteration. The implication of this is that the changes in the Diapason calculation that is performed in step 2 of the next iteration will also be negligible, which will result to even smaller changes in SM_{interp} during step 3, and so on. Eventually, by iteration 3, as shown in Fig. (**11**), the changes are so small as to be considered negligible for all practical purposes. Note, also, Fig. (**11**) shows the value fluctuations occurring in a particular emotion attribute (emotion attribute 4 ("Fear") for a particular K-line (K-line 4). We select this data because it exhibits the most vivid fluctuations across the K-line collection. All other emotion attributes and other K-lines exhibit even smaller fluctuations.

Based on the above discussion, all K-nodes that point to the same song have the same Diapason value by the end of the algorithm Part B. At this point, the K-line mesh is considered to be formed, and the system is ready for use.

E-Stage II: Using the System

In this stage the user enters her SM at the beginning of each session and as a result the system provides her with a playlist. In our testing we set the length of the produced playlist to be exactly 10 songs. Each user performs 10 sessions and in each session the user listens to ten songs. During the listening process, the user evaluates each of the 10 songs on a scale from 0 to 10. Ten means that in the user's opinion, the song matches her mood perfectly; zero means that the song does not match her mood at all.

In addition to judging how a user perceives the quality of the retrieved songs, we design our experiment in such a way that our retrieval mechanism is also compared to the random song selection mode which is typically used when operating a song listening devise. In doing so, each of the 10 playlists presented to each evaluator is a *mix* of two types of songs. One type of songs are the songs retrieved by our method, based on the navigation of the K-line mesh, as described in section 2. The second type of songs in the mix, are songs randomly selected from the songs of the K-line mesh. Each of the playlists presented to the evaluator consists of 10 songs. Of those 10 songs, 5 are selected by our retrieval mechanism and 5 are selected randomly from the song repository. To ensure that the relative locations of the random songs with respect to the system selected songs is neutral toward forming any bias to the user's SM during the evaluation process, we create the 10 playlists by mixing the randomly selected songs and the system selected songs in three different ways:

- **Mix (I).** From a playlist of 10 songs which are selected by the system, we randomly remove 5 songs and replace them with 5 randomly selected songs.

- **Mix (II).** From a playlist of 10 songs which are selected by the system, we remove the *last* 5 songs and replace them with 5 randomly selected songs.
- **Mix (III).** From a playlist of 10 songs which are selected by the system, we remove the *first* 5 songs and replace them with 5 randomly selected songs.

Using the above three mixing strategies, we create 10 playlists: 4 of the 10 playlists are of type Mix (I); 3 of the 10 playlists are of type Mix (II); 3 of the 10 playlists are of type Mix (III). Before the playlists are presented to the user for evaluation, we put the playlists in random order so that the user does not intentionally receive blocks of playlists of the same mix.

Performance Comparison

As the users listen and evaluate the songs, we record the user-reported satisfaction values for each song. Table **1** shows relevant statistics from the reported values.

Table 1.

	Our Algorithm	Random Selection
Mix (I)	72.6	22.2
Mix (II)	74.5	21.7
Mix (III)	69.9	22.9
Mix (ALL)	72.3	22.3

In Table **1**, each of the entries of the first three rows shows the average of all reported user satisfaction values (across all participants and across all playlists) for all songs S, such that S is a song which was selected by either our algorithm ("Our Algorithm" column of the Table), or randomly from the K-line mesh ("Random selection" column of the Table). For example, **[Mix (III), Random selection]** is the average of all reported user satisfaction values (across all participants and across all playlists) for all songs S, such that S is a song which was selected randomly from the K-line mesh and S is in a playlist formed with Mixing strategy (III). And so on. The last row of the Table **1** shows the average of all reported user satisfaction values (across all participants and across all playlists) for all songs S, such that S is a song which was selected by our algorithm and S is in any playlist regardless of the mixing strategy used to form that playlist. Essentially, the values of row MIX(ALL) are the averages of the values of the entries of the rows MIX (I), MIX (II), and MIX (III).

Conclusions from the Performance Results

Behaviour of our Algorithm for the Formation of the K-Line Mesh: The essence of the results shown in Figs. (**10**) and (**11**) is that the iterative process of Algorithm Part B stabilizes the Diapasons and converges, within a small number of iterations. As illustrated by Fig. (**10**), during the execution of step 2 of algorithm Part B, the *Diapasons stabilize across the entire K-line collection*. As illustrated by Fig. (**11**), during the execution of step 3 of algorithm Part B, the SM_{interp} also stabilizes within a few iterations. This means that algorithm B overall forms a stable K-line mesh.

Quality of the K-Line Mesh Navigation Algorithm Part C and Comparison of our Method Against the Most Frequently Used Typical Alternative of Random Song Selections: For Mix (I) the overall average for our algorithm is 72.6%. For Mix (II) the overall average is 74.5% and for Mix (III) the average for our algorithm is 69.9%. For Random song selection the average is approximately the same for all three Mixing strategies (with an insignificant difference of no more than 1.2%). This is strong evidence that random song selection performs equally in all cases. Therefore, our different mix strategies do not generate any confusion to the user regarding what constitutes a randomly selected song versus a song that is supposed to conform to her SM. For Mix (ALL), the average satisfaction among our algorithm's song selection is 72.3% and among Random song selections is 22.3%. The overall percentage is calculated by taking the overall average among all playlists and among all users. Since there are eight users and each user evaluates 10 playlists, there is a total of 80 playlists containing a total of 800 songs (recall, each playlist has 10 songs). Note, our repository contains 300 songs, so there are many duplicates among the 800 songs. However, there are no duplicates within each playlist. Of those 800 songs, 400 are songs retrieved by our algorithm Part C, and 400 are songs selected randomly for the purposes of creating the different types of mix. Our observed 22.3% average for randomly selected songs is in agreement with the typical reported 20% average for randomly generated playlists as, for example, is reported in [3]. We attribute the 2.3% difference (our 22.3% versus the generally reported 20%) to be either coincidental or due to the fact that in our experiments we deliver the randomly selected songs *together* with songs that are generated by our algorithm (using our mix strategies). We speculate that as the user listens to SM-conforming songs (which are included in the delivered playlists of different types of mix), her satisfaction posture is positively affected (such influences to a person's mood are reported in [30]) and as a side-effect the user gives also higher satisfaction level points to the randomly selected songs. In either case, the 2.3% difference is not significant.

The results of the evaluation clearly indicate that the proposed method is much more preferable than the random song selection which is the common choice of a typical music listener. Specifically, the results of the evaluation (72.3% versus 22.3%) suggest that our method outperforms the random song selection method by a margin of more than 3 times.

4. CONCLUSION

Our method is comprised of two main parts. The first part is algorithm Part A together with Algorithm Part B. The second part is algorithm Part C. All algorithms are described in Section 2. Algorithms Part A and Part B organize a song repository upon recording a user's memory experiences from previous listening activities. Algorithm Part C provides a retrieval mechanism acting on the organized repository. The

user's memory experiences are captured in K-lines, an AI-related data structure proposed by Minsky [8]. Our retrieval mechanism is based on activating appropriate K-lines from a formed K-line mesh and is inspired by Minsky's theory of memory as described in [6-8].

We implement our method and test it by monitoring the internals of our algorithm execution and by recording, at the same time, feedback of human participants. The results of the introspection of the algorithms' execution reveal that our algorithm converges within a few iterations and forms a stable K-line mesh (Figs. **10**, **11**). The feedback of the human participants (Table **1**) provides clear evidence that our method outperforms the typical random song selection mode by a margin of more than 3 times.

Our method is implemented here for a song collection but it is applicable to any type of media, such as video clips, images, and text. This is because our method relies on the "overall impression" and "degree of likeness" of the consumer of the media (as represented by the consumer's SM), rather than on characteristics of the media content which are typically proprietary to the type of the media. To the best of our knowledge, there is no published research work that uses K-lines for organizing any type of media. Moreover, as reported in [36], "*… no implementation details have been given [by Minsky], and in twenty-five years no one has claimed to have implemented it [K-lines]*". [36], end of section 2.1, page 13. ([36] discusses a topic completely different from media organization and chooses an approach other than K-lines).

FURTHER RESEARCH DIRECTIONS

The work presented in this paper is amenable to further research. We consider the following research directions worth further investigation.

Analytical Performance Evaluation of Our Method. In this work we evaluated our proposed method using human participants. Although this is a common way for evaluation of such systems, we are interested to provide an analytical performance evaluation as well. The main difficulty of such an evaluation is the development of a model which captures the expected song selections for constructing the initial K-lines. In other words, the issue is how to model a system-driven, as opposed to a human-driven, initial song selection and how to model the periodic input of an appropriate SM which is required during the K-line formation process. Note, the obvious approach of forming K-lines by randomly generating SMs and pairing them with randomly selected songs will provide a working system. However, any results generated by such a system will be meaningless. This because such a system contains only artificial (and, in no way meaningful) pairings of the SM_{user} and the Song components of any K-node.

Refinement of SM Representation. In our model we use the 6 basic Ekman emotions to represent a SM. Although Ekman's emotion theory is widely accepted and often used in similar projects, it is of interest to explore how different representations of the SM affect the system. Specifically, Minsky's mind architecture [6, 7], proposes that a state of mind is an instance of a collection of "agents" that are active at any given time point. It is of great interest to explore representations of SM based on this architecture. The main problem in such an approach is to map emotional states to agent activation configurations. To the best of our knowledge no such research has been reported.

Mood Altering Effects of the Listening Process, when the System is Used: As it stands, the system is trained and the K-lines are formed based on algorithms Part A and Part B, as described in Section 2. At the end of algorithm Part B, the K-lines mesh is *frozen* (i.e., no further modified), although it allows for intelligent song selection, as done by algorithm Part C. It is conceivable (and probably likely, according to the discussion in subsection 2.1) that when we use the system, the process of listening to the songs offered by the system further alters the mood of the listener and in ways that were not possible to be predicted during the K-line mesh formation. As such, the act of using the system may cause the user to fall into new SMs which are different from the SMs that are predicted by the system through its K-line mesh. Although those new SMs may only slightly differ from the SMs that are predicted by the K-line mesh, it is worthwhile to investigate the significance (and possible consequences) of such variations. Such an investigation could produce research in at least two different fronts:

1. How to amend the system in order to be able to adjust in the presence of such variations. Possible scenarios could include dynamic updates of the K-lines, or the development of some indexing or clustering techniques that take in account the dynamic mood variations.

2. How to use the knowledge of such mood variations, in conjunction with the songs that are available in the K-line mesh, in order to build a system with *mood-altering capabilities*. In such a system, the primary feature would be that a user enters a desired target SM and the system provides a series of songs whose purpose is to *influence* the user's mood (rather than conform to the user's mood, as is done by the current system) so that eventually reaches the desired target input SM.

Testing the Current System with X-Dimensional SMs, for X < 6; Automation of the Input of the User SM: In our current system we use all six Ekman's emotions to express a SM. It may be of interest to see if there are any significant findings if only fewer than six emotion attributes are active and the remaining attributes are dormant. If such a system works as well as the current system, the obvious benefits are simplification and increased ease of use. In addition, a benefit may also be, surprisingly, increased accuracy. According to [16] and [17] the average person does not possess enough self-knowledge and self-awareness to know exactly what her SM is at any given point. The problem may be further complicated in cases that cognitive clarity is affected by increased levels of anxiety or fatigue. In case that the user is required to input fewer than six values for the system to form a SM, it might be that the accuracy of the user in expressing her true SM, is increased. As a result, the performance of the

system, in terms of providing satisfactory playlists, will also be increased. Similar effects are possible if the process of acquisition of the user's SM is automated. Several automatic emotion acquisition mechanisms exist as, for example, the Electroencephalography (EEG) devises, and the devise described in [37]. Our future plans include experimenting with some of these devises.

REFERENCES

[1] H. Mandviwala, S. Blackwell, C. Weikart, and J. Van Thong, "Multimedia content analysis and indexing: evaluation of a distributed and scalable architecture", *ITCom, Internet Multimedia Management Systems IV*, vol. 5242, pp. 137-145, 2003.

[2] L. Rutledge, J. van Ossenbruggen, and L. Hardman, "Structuring and presenting annotated media repositories", *Proceedings of the 13th International World Wide Web conference on Alternate track papers & posters*, ACM press, 2004, pp. 466-467.

[3] G.T. Elliott and B. Tomlinson, "Personal Soundtrack: Context-aware playlists that adapt to user pace", *Proc. CHI 2006*, Canada, ACM Press, pp. 736-741, 2006.

[4] S. Dornbush, Jesse English, Tim Oates, Zary Segall, and Joshi Anumpam, "XPod: A Human Activity Aware Learning Mobile Music Player", *Proc. Workshop on Ambient Intelligence, 20th International Joint Conference on Artificial Intelligence*, 2007.

[5] P. Ekman, "Facial expression of emotion", *American Psychologist*, vol. 48, pp. 384-392, 1993.

[6] M. Minsky, *The emotion machine*, Simon & Schuster, November 7, 2006.

[7] M. Minsky, *The Society of Mind*, Simon & Schuster, March 15, 1988.

[8] M. Minsky, "K-Lines: A Theory of Memory", *Cognitive Science*, vol. 4, pp. 117-133, 1980.

[9] R.W. Picard, *Affective Computing*, The MIT Press, July 31, 2000.

[10] B.R. Duffy, "Fundamental Issues in Affective Intelligent Social Machines", *Open Artificial Intelligence Journal*, vol. 2, pp. 21-34, 2008.

[11] R.W. Picard, "Toward computers that recognize and respond to user emotion", *IBM Systems Journal*, vol. 39, pp. 705-719, 2000.

[12] D.C. Dennett, *Consciousness Explained*. New York: Little, Brown and Company, 1991.

[13] C.D. Pope, "Somatic Computationalism: Damasio's clever error", M.A.Thesis, Louisiana State University, Agricultural and Mechanical College, August 2007.

[14] A. M. Isen, K. A. Daubman, and G. P., Nowicki "Positive affect facilitates creative problem solving", *Journal of Personality and Social Psychology*, vol. 52, pp. 1122-31, June 1987.

[15] M. Minsky, "Music, Mind, and Meaning", *MIT-AI Laboratory Memo 616*, fall 1981, http://web.media.mit.edu/~minsky/papers/MusicMindMeaning.html (accessed August 23, 2008).

[16] D. Goleman, *Emotional intelligence*, Bantam Books, 1995.

[17] D. Goleman, *Social Intelligence*, Bantam Books, 2006.

[18] A. Damasio, *Descartes' Error: Emotion, Reason and the Human Brain*. New York: Penguin Books, 1994.

[19] A. Damasio, *The Feeling of What Happens: Body and Emotions in the Making of Consciousness*. New York: Harcourt, 1999.

[20] H. Liu, H. Lieberman, and T. Selker, "A Model of Textual Affect Sensing using Real-World Knowledge", *Proceedings of the 8th international conference on Intelligent User Interfaces, IUI 2003*, Miami, FL, USA, 2003, pp.125-132.

[21] K. Matsumoto, F. Ren, S. Kuroiwa, and S. Tsuchiya, "Emotion Estimation Algorithm Based on Interpersonal Emotion Included in Emotional Dialogue Sentences", *MICAI 2007: Advances in Artificial Intelligence*, Lecture Notes in Computer Science, vol. 4827, Springer, 2007, pp. 1035-1045.

[22] P. Singh, "Failure Directed Reformulation", M.Eng.. thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 1998.

[23] P. Singh, "*EM-ONE":* "An Architecture for Reflective Commonsense Thinking", Ph.D. Thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, June 2005.

[24] P. Singh and M. Minsky, "An Architecture for Combining Ways to Think", *International Conference on Knowledge Intensive Multi-Agent Systems*, Cambridge, MA, 2003.

[25] N. Hawes, J. Wyatt, and A. Sloman, "An Architecture Schema for Embodied Cognitive Systems", *Technical Report COSY-TR-0610*, November 24, 2006, (accessed August 21, 2008). http://www.cs.bham.ac.uk/research/projects/cosy/papers/#tr0610

[26] A. Sloman, "Grand challenge 5: The Architecture of Brain and Mind: Integrating Low-Level Neuronal Brain Processes with High-Level Cognitive Behaviours in a Functioning Robot", *Technical Report COSY-TR-0607*, July 2006, (accessed August 21, 2008). http://www.cs.bham.ac.uk/research/projects/cosy/papers/#tr 0607

[27] M. Minsky, P. Singh, and A. Sloman, "The St. Thomas Common Sense Symposium: Designing Architectures for Human-Level Intelligence", *AI Magazine*, American Association for Artificial Intelligence, pp. 113-124, Summer 2004.

[28] M. Minsky, "A Framework for Representing Knowledge-FRAMES", *MIT-AI Laboratory Memo 306*, June, 1974, (accessed August 21, 2008). http://web.media.mit.edu/~minsky/papers/Frames/frames.html

[29] A.A. Toptsis and A. Dubitski, Organization and Retrieval in Affectively Annotated K-line Indexed Media Repositories, *Proc. Software Engineering and Applications* (SEA 2008), pp. 148-153, November 2008.

[30] K.R. Scherer, and M.R. Zentner, Emotional Effects of Music: Production Rules, *Music and emotion: theory and research*, Oxford University Press, pp. 361-392, 2001.

[31] O. Bretscher, *Linear Algebra With Applications, 3rd ed.*. Upper Saddle River NJ: Prentice Hall, 1995.

[32] C. R. Rao, H. Toutenburg, A. Fieger, C. Heumann, T. Nittner, and S. Scheid, *Linear Models: Least Squares and Alternatives*, Springer Series in Statistics, 1999.

[33] T. Kariya, H. Kurata, *Generalized Least Squares*, Wiley, 2004.

[34] J. Wolberg, *Data Analysis Using the Method of Least Squares: Extracting the Most Information from Experiments*, Springer, 2005.

[35] H. Liu, and P. Maes, "What would they think? a computational model of attitudes", *Proceedings of the ACM International Conference on Intelligent User Interfaces, IUI 2004*, pp. 38-45, Portugal, 2004

[36] C. Miller, "Modeling Trust in Human Conversation", M.Sc. Thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2006.

[37] C. Peter, E. Ebert, and H. Beikirch, "A wearable multi-sensor system for mobile acquisition of emotion-related physiological data", *Proceedings of the First International Conference on Affective Computing and Intelligent Interaction: ACII 2005*, Springer, pp. 691-698, 2005.

Received: September 2, 2008 Revised: November 10, 2008 Accepted: November 17, 2008