18551 : FINAL REPORT

Group-5 Spring 2011

MOOD GENIE

Song Emotion Identification on a DSK

Swetha Chigurupati (schiguru@andrew.cmu.edu)

Zhi Yang Lim (zlim@andrew.cmu.edu)

Chun How Tan (chunhowt@andrew.cmu.edu)

Content:

1. Introduction

2. Problem

3. Novelty

4. Solution Overview

5. Dataset

6. Feature Extractions

7. Emotion Classification using SVR

8. Lab Demos

9. DSK vs. PC breakdown

10. Speed and memory on DSK

11. Results

12. Future Works

13. Semester Schedule

References

1. Introduction:

Songs are the embodiment of the expression of the human emotion. Often times, we play songs to reflect the mood that we are in. However, it can be difficult to identify the songs with the right emotion to play in a particular situation given a large song library. We propose to create an application that will classify songs based on its emotion content using techniques from signal processing and pattern recognition.

2. Problem:

The problem we are proposing to solve is – given a studio-recorded song (i.e. noiseless), we classify its emotion to be one of 4 categories of emotion, following the Juslin's theory along with Thayer's emotion model[1]. The 4 emotions that we proposed to classify are "happy", "sad", "angry", and "relaxed". Thayer's 2-dimensional emotion model uses arousal, which represents the energy of the song, for y-axis, and the valence, which represents the stress of the song, for x-axis. Fig 1 is the diagram of how each emotion is classified on the 2-D plane.

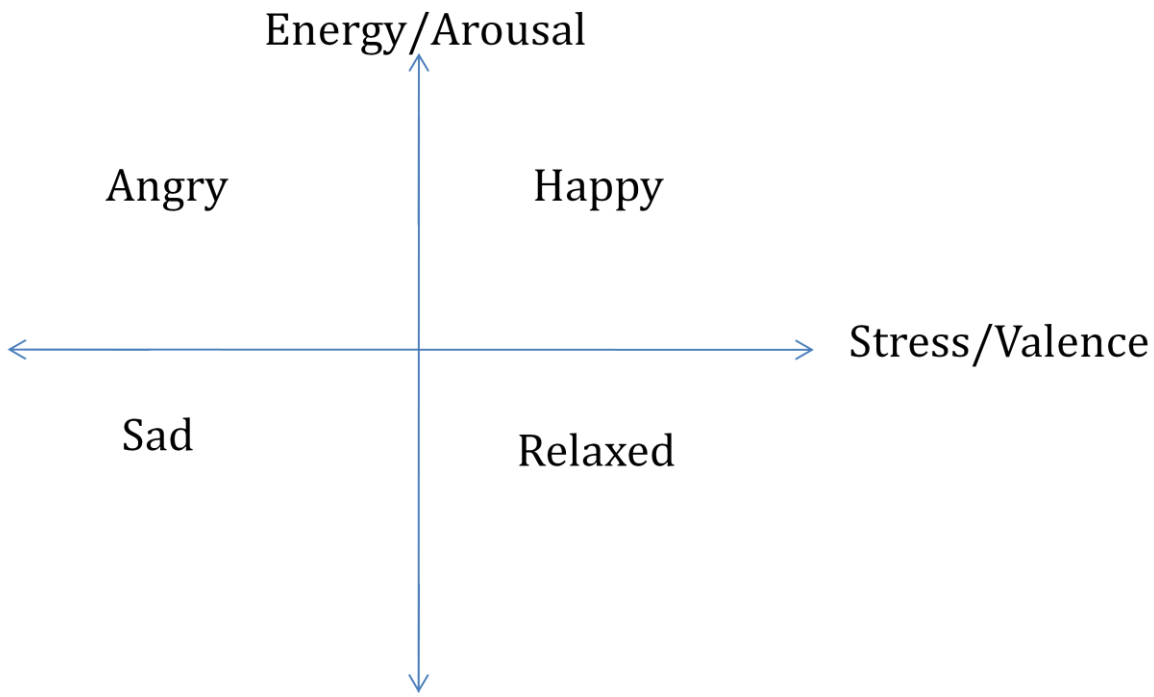

Fig 1. Thayer's 2-D emotion model.

3. Novelty:

Commercial version for the emotional classification of songs exists currently (eg. SensMe by Sony Ericsson), but the algorithms used for these commercial software are proprietary. In addition, all these commercial applications are either run on the PC or do all their processing on the PC, then push the results to the mobile devices, in the case of SensMe.

To the best of our knowledge, this procedure has never been implemented on a DSK. Our goal is to implement the classification algorithms on the DSK. There are a lot of issues to be resolved, including network transfer and memory storage because of DSK's limited processing power. In the context of 18551, this category of projects has not been attempted before, and thus represents a new frontier in which to take the potential usages of a DSK. All other audio projects done so far in 18551 have not dealt in analyzing the emotional aspects of audio.

4. Solution Overview:

Our process can be split into the training and demo part.

4.1. Training

For training, we need to ensure homogenous and standardized datasets. To achieve that, we first normalized the songs to 89 dB using the software MP3Gain [2], and converted the mp3 songs into wav format using the matlab routine, mp3read [3]. Then, we made all the songs mono, with the same bit rate of 128 kbps and same sampling frequency of 44.1 kHz using the same mp3read routine. Next, we manually chose a 60 seconds frame for each song that has the most interesting content inside, such as the chorus part. Refer to section 5 for more information about the standardization process.

From the standardized songs, we extract 4 representative song features from them. The features are pitch, loudness, beats and spectral centroid. Next, we generate from these 4 features, 8 variations of values – average energy, standard deviation of energy, average fundamental frequency, standard deviation of fundamental frequency, number of fundamental frequencies higher than 0.25 * maximum fundamental frequency, 75 percentile beats, average spectral centroid and standard deviation of centroid.

Using these 8 features, we brute-force all 255 combinations of them in the SVR training. Since SVR returns a real value, we decided to separate the SVR training on the arousal and valence. In general, for X some subset of 8 features above, we have 2 independent mapping functions f: X -> O, where O $\in$ R is the valence/arousal of the song. Then, we chose the best combination of features for valence and best combination

of features for arousal. Finally, we trained SVR models based on these combinations and store them for the second part – demo.

Here is the flow chart showing our training process:

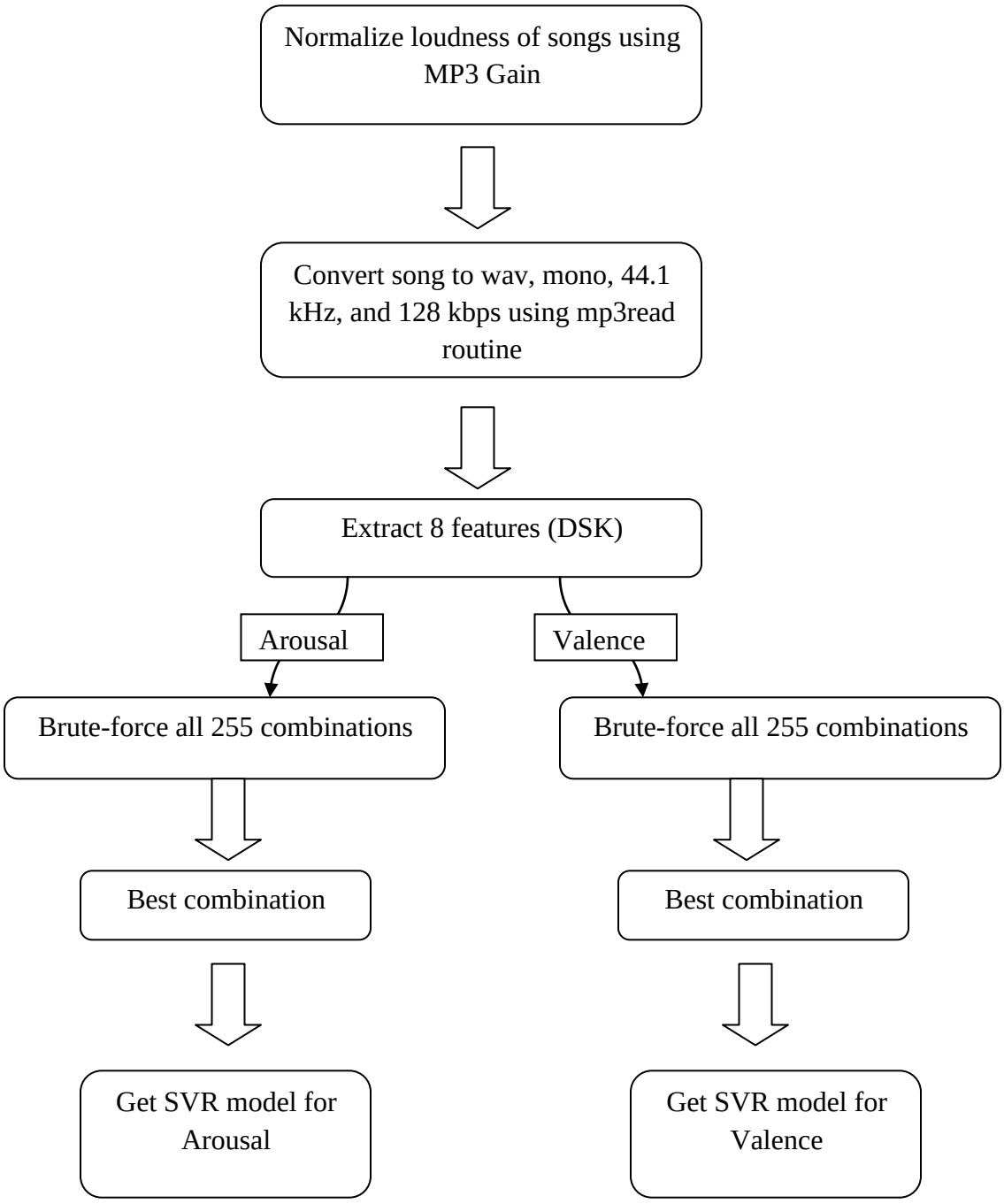

Fig 2. Flowchart of the training process.

All parts done on PC unless otherwise stated.

4.2. Demo

Step 0: Preprocess the song by converting into wav format. Note that, the song need not be standardized as in training data. Songs of arbitrary length, arbitrary bit rates and loudness could be fed into our system. There are no clear distinctions of the results between standardized songs and arbitrary songs. Refer to Section 11 for more information on our results.

Step 1: We first loaded the SVR models found for arousal and valence from the training part into the DSK.

Step 2: We fed 30 seconds at each time into DSK for feature extractions. The feature extraction was done on each second, and so we have 30 values per feature.

Step 3: Then, we computed the 8 features based on these 30 values and stored these features on DSK (averaging with old values if this is not the first 30 seconds window).

Step 4: If there are more frames of songs, we push them into DSK again and repeat Step 2. Else, we go to Step 5.

Step 5: Using the averaged features, we use the SVR model stored to compute the arousal and valence on DSK.

Step 6: We sent the arousal and valence back to PC for displaying on the GUI.

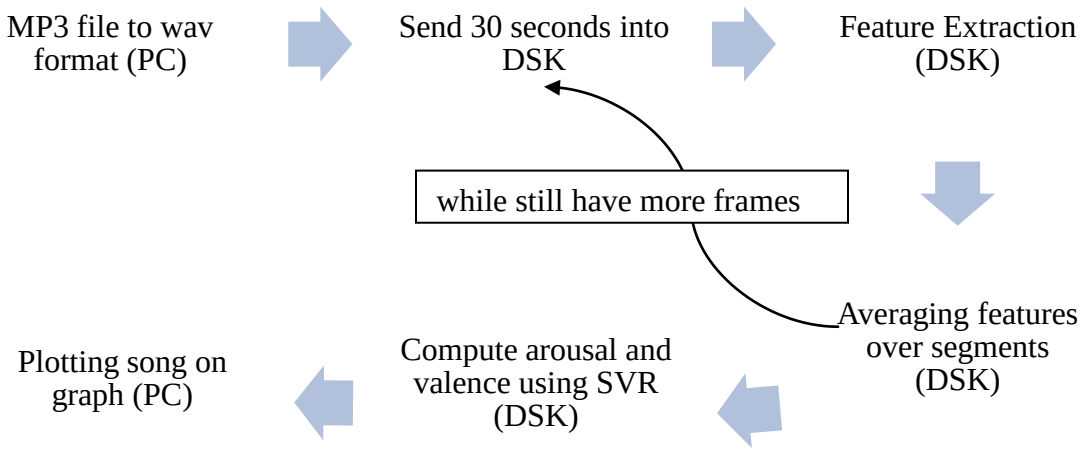

Fig 3. Flowchart on the final demo system.

5. Datasets:

We used 71 English songs for the whole project which includes 20 angry songs, 18 happy songs, 16 relaxed songs and 17 sad songs. Out of these 71 songs, 40 songs were used for training data, 15 songs for development data (used for selecting features, refer to Section 7 for more information) and 16 songs for testing data. Refer to Appendix 1 for the list of songs. The training, development and testing are chosen alphabetically, i.e. the first 40 songs are training, next 15 are development etc.

For training, we need to ensure homogenous and standardized datasets. To achieve that, we first normalize the songs to 89 dB using the software MP3Gain [2]. According to the software's description, the software first uses the ReplayGain algorithm to calculate the loudness of the song. Then, the software does loudness normalization. The software adjusts and normalizes the songs so that human will perceive the songs to sound the same loudness, instead of doing simple peak normalization.

Then, we convert the mp3 songs into wav format using the matlab routine, mp3read [3]. Then, we make all the songs mono with the same bit rate of 128 kbps and same sampling frequency of 44.1 kHz using the same mp3read routine.

Next, we manually chose a 60 seconds frame for each song that has the most interesting content inside, such as the chorus part.

After we have this list of 71 songs, all the three members independently and individually annotate the arousal and valence of each song clip. Finally, we take the average of valences and arousals across the 3 members to be considered as the ground truths.

6. Feature Extractions:

Based upon the paper in which Professor Roger B. Dannenberg of CMU submitted to the 10th International Society for Music Information Retrieval Conference [1], we extracted the following features from the segment:

Feature	Explanation
Average Energy	The average energy in a song is an important measure for quantifying the loudness in a song. Also, the regularity of music can be measured by the standard deviation of average energy. The song is separated into number of frames and the average energy in each frame is calculated. For every frame in the song, 1. The average energy is calculated as, Average Energy (x) = AE(x) = $\frac{1}{N}\sum_{t=0}^{N} x(t)^2$ 2. The standard deviation of average energy is calculated as, Standard deviation of AE = σ (AE(x)) = $\sqrt{(\frac{1}{N}\sum_{t=0}^{N}(AE(x) - x(t)^2))}$ Where, x is an input frame, t is the time in samples, and N is the length of x in samples. The Average Energy and Standard Deviation of Average Energy are used to represent the arousal attribute of a song.
Fundamental Frequency[4]	Pitch is the perceived fundamental frequency of the sound. Many modes of vibration occur simultaneously, the vibration that has the slowest rate is called the fundamental frequency. As such, this feature simply involves extracting out what the fundamental frequency of the song is. The first frame has been discarded to account for noise in the song. The pitch in each other frame is considered for classification. For every frame in the song, 1. The auto-correlation output is computed. 2. Depending on the above output, the zero crossings in the signal are identified. Zero crossing is where the signal changes its sign. 3. Depending on the zero crossing rate, the frequency corresponding to the sample with higher slope is determined which represents the fundamental frequency. The average fundamental frequency and the standard deviation of the fundamental frequency are considered along with the count representing

	number of fundamental frequencies which are higher than the average fundamental frequency of the frames considered.
Spectral Centroid[4]	The spectral centroid is a measure used to indicate where the "center of mass" of the spectrum is. It is calculated as the weighted mean of the frequencies present in the signal, determined using a Fourier transform, with their magnitudes as the weights. The Hamming window is used for obtaining centroid in each frame of the entire song. The song is divided into frames and the centroid of each frame in the song is calculated. For every frame in the song: 1. The center frequency and the weighted frequency value in each bin are determined. 2. Using the center frequencies and weighted frequency values, the centroid of each bin is calculated using the following formula. $$Centroid = \frac{\sum_{n=0}^{N-1} f(n)\, x(n)}{\sum_{n=0}^{N-1} x(n)}$$ where *x(n)* represents the weighted frequency value, or magnitude, of bin number *n*, and *f(n)* represents the center frequency of that bin. Then the average and standard deviation of the centroid values in all the frames are calculated and used for classification.
Beats[5]	In designing the beat algorithm, we must take into account our algorithm must be able to handle all types of songs, including noisy songs such as rock or pop music. This immediately rules out the use of simply sound energy algorithms to detect beats. Therefore, the algorithm must have the ability to determine on which frequency subband we have a beat and if it is powerful enough to take it into account. Therefore, what we want to do is try to detect big sound energy variations in particular frequency subbands. According to the Parseval Theorem, the energy computing in the time domain is the same as the energy computed in the frequency domain, so there should not be any difference between computing the energy in the time domain or the frequency domain. **For every 1024 samples:** 1. Compute the FFT of the 1024 new samples taken in. 2. From the FFT we compute the 1024 frequency amplitudes of our signal by taking the square root of the sum of squares.

	3. Divide the buffer into 16 subbands, and compute the energy on each of these subbands and store it at Es. Thus Es will be 16 sized and Es[i] will contain the energy of subband 'i': $Es[i] = \frac{64}{1024} \sum_{k=i*64}^{(i+1)64} Ampltiude[k]$ 4. Now, we have an energy history buffer called Ei that corresponds to each subband 'i'. This buffer contains the last 44 energy copmutations for the 'i' subbands. We compute the average energy Ei for the 'i' subband simply by using: $Ei[k] = \frac{1}{44} \sum_{k=0}^{43} Ei[k]$ 5. If the buffer is full already (i.e. we already went through 1024 * 44 samples), we do a memmove to shift the sound energy history buffers Ei by 1 index. 6. Pile in the new energy value of subband 'i' $Ei[0] = Es[i]$ 7. For each subband 'i' if Es[i] > Ei we have a beat! 8. If we have a beat, we then find the beat by taking the largest amplitude in that subband and then converting it to frequency to obtain the beat value. We chose to use the 75 percentile of the beats for each frame.

7. Emotion Classification using SVR

7.1. SVR Overview

We perform a supervised SVR training using our dataset of emotion annotations as detailed in the datasets section at section 5. We used nu-SVR.

The basic idea for SVR is given a feature vector X, we want to find parameters w and b, where w is a vector of same size as X and b is a scalar, such that we will obtain a linear mapping function f(X) where:

$$f(X) = w^T X + b$$

To obtain w, we want to solve a quadratic optimization problem to minimize the norm of w^2 [6].

$$minimize\ \frac{1}{2}||w||^2 - C\left(v\varepsilon + \frac{1}{N}\sum_{i=1}^{n}(\varepsilon + \varepsilon_i^*)\right)$$

$$where\begin{cases} y_i - w^T X_i - b \le \varepsilon + \varepsilon_i \\ w^T X_i + b - y_i \le \varepsilon + \varepsilon_i^* \\ \varepsilon_i, \quad \varepsilon_i^* \ge 0 \end{cases}$$

Here, ε_i^* and ε_i are two slack variables to allow some misclassification to happen rather than always require to have 100% classification accuracy, which might be impossible on certain data where we cannot split the classes nicely.

To obtain b, we can use Karush-Kuhn-Tucker (KKT) conditions [6].

We can also expand our regression mapping to nonlinear functions using Kernel function. The idea to map values to higher order is because if some feature vectors are not separable at the lower order, projecting them into higher order might allow them to be classifiable. One of the most popular and useful kernel functions is the Radial Basis Function (RBF), defined as [6]:

$$\emptyset = \exp\left(-\gamma|x_i - x_j|^2\right)$$

$$where\ \gamma\ is\ one\ of\ the\ parameter\ to\ be\ chosen.$$

The whole SVR training may be complicated and become nasty as we tackle non-linear mapping. Fortunately, we found LIBSVM, a library for SVR written in C++ and Java online [7]. This library supports good interface to do training with SVR.

7.2. Training Approach

LIBSVM supports multiple SVM and SVR implementations, and multiple kernels. We used nu-SVR and RBF function for this project.

During training, each trial consists of doing cross-validation on the training data to choose the best parameters (such as epsilon, gamma and cost in the equations above). Once the best parameters that maximize the cross-validation were chosen, we run the SVR on the development data to see the accuracy rates. We ran 255 trials each for arousal and valence using all the possible combinations of features (8 features => $2^8 - 1$ (empty subset) = 255 combinations) and chose the feature set that maximize the cross-validation rates and break ties by the accuracy on the development data.

Training results can be found in Section 11.

8. Lab Demos

8.1. Lab Demo 1: Matlab's Mex Interaction with DSK

This program allows us to natively interact with the DSK in order to parse the emotion of the chosen song. The interaction of the Matlab GUI with the DSK is facilitated by a self-written Matlab Mex (Mtlab EXtension) function written in C. This facilitates the portability of our program and would potentially allow us to compile the program to be used across all PC platforms.

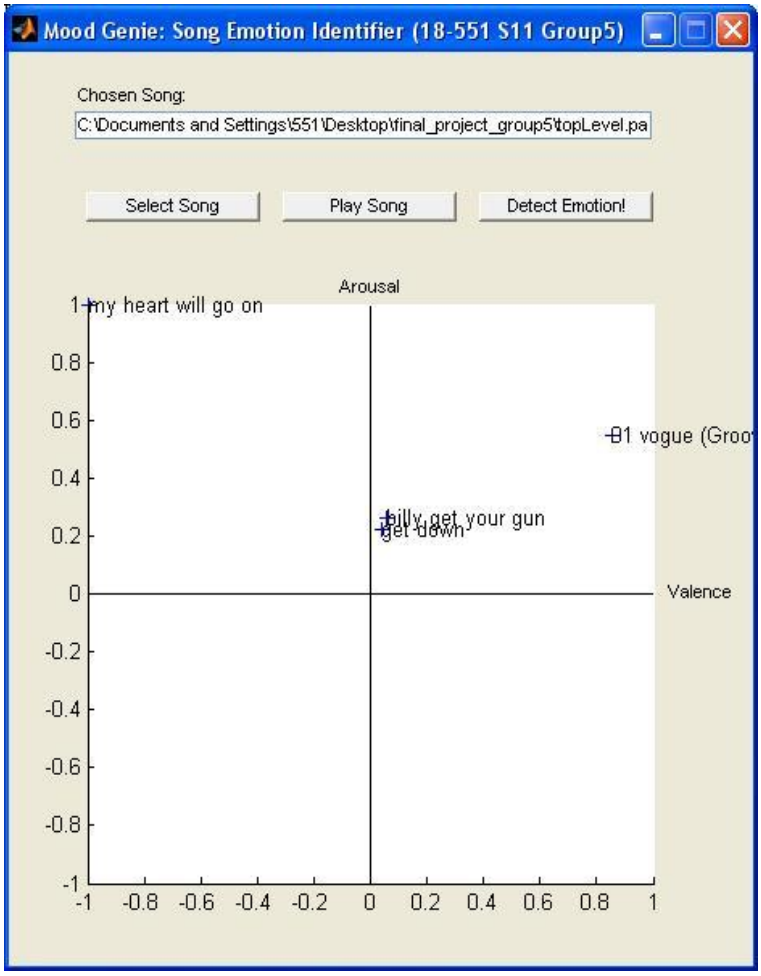

Fig. 4. Lab Demo 1: Matlab's Mex Interation with DSK

Chosen Song: This list the song that have been selected via the 'Select Song' button or typed into the text box by the user.

Select Song: This allows the user to select the song that he/she wants to see the emotion for.

Play Song: This plays the chosen song using native Matlab functions. This allows the user to play the song for himself and have a general feel as to where the song might

fall on the graph. There is an in-built error checker that prevents anything but .mp3 file extensions from being played in order to avoid crashing the program by erroneous user input. This button changes to "**Stop Song**" when a song is being played to allow the user to stop the song being played at any point in time.

Detect Emotion!: This passes the song into the DSK. Details of this procedure are described later on in this section. Once again, before the song is passed to the DSK, there is an in-built error checker that prevents anything but .mp3 file extensions from being passed into the DSK in order to avoid crashing the program by erroneous user input.

Graph: This displays the emotion of the song once it has been determined by the algorithm on the DSK. The song's location is marked by a blue 'x' with the song title placed next to it to allow easier reading of the graph once multiple songs' emotions have been detected.

Once the "**Detect Emotion!**" button has been pressed, after the error checker module has been passed, the entire path of the chosen song is passed into the Matlab Mex C function. Once there, a busy dialog comes up on the screen that prevents the user from clicking any more buttons on the GUI. In the Mex C function, it connects to the DSK via TCP and establishes a connection. Once this happens, the song is then passed into the DSK in 30 second segments. The Mex C function continues to do this until the DSK has called for all segments of the song, and then waits for the DSK to output a singular 2 value struct containing the final values for arousal and valence in order to plot it on the graph of the GUI. This value is then passed back to the GUI for plotting.

Our program is designed to be robust and thus would be able to take in any length of mp3 formatted song. In testing, it was able to successfully output the arousal and valence for 6 minutes long songs. The performance time of the DSK algorithm is directly proportional to the length of the input song (8 seconds for every 60 seconds of song). However, we do not normalize any of the songs that are fed into the DSK, thus if a song was ripped at a different dB level compared to our standard dB level (89 dB), the accuracy of the results may be affected. We do not believe this accuracy to be adversely affected though, as our own annotations of these songs agree with the output given by our system.

One interesting initial bug of note in our program is the fact that we have to sleep the program for a very brief period of time before we allow any data to transfer to the DSK (this was eventually set at 300 milliseconds). While the root cause of this is not exactly known, we suspect that it is due to a lag time that requires the TCP connection to properly establish the three-way handshake. This sleep time is not noticeable to the end user, but nonetheless represents an interesting bug.

8.2. Lab Demo 2: Java demo on Processed songs

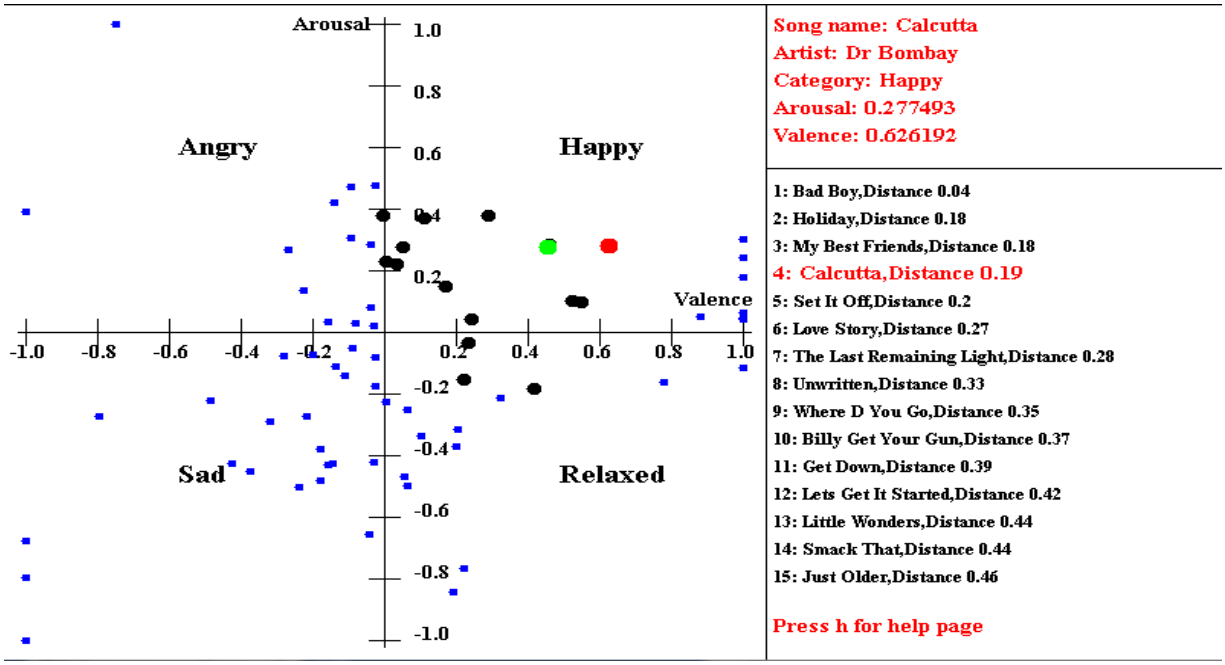

Fig 5. Lab Demo 2: Java demo on processed songs

The second lab demo implemented in Java is developed for the end-users purpose. This program takes in a labeled list of songs, with the corresponding artists, arousal and valence value. Then, the user can use this program to pick the "emotion" they want and the program will play the songs that have the most similar emotions to the emotion level chosen. The functionality to play MP3 in Java is obtained from an implementation in [8] and the functionality to accept mouse and keyboard inputs from the users is credited to David Kosbie in [9].

A few functionalities provided in this demo are:

1. Users can select any point on the plot and click it. Then, the top 15 nearest songs will be shown on the panel on the right and the Java program will play the first song. User can change the song they want the Java program to play by pressing up or down key on the keyboard.
2. On the plot, the blue dots show all the emotion values for every sons in the database accessible to this demo program. Pressing key "a" will switch between showing the blue dots and not showing them.
3. The black dots on the plot are the dots representing the top 15 songs closest to the point chosen (shown as green dot). The red dot represents the location of the current song being played.

9. DSK vs. PC Breakdown

PC:

- Standardization of songs (loudness normalization, conversion to wav, mono, 128 kbps and 44.1 kHz) for training
- Conversion from mp3 format to wav format for demoing
- Take in the input choice of song from the user
- Display the arousal and valence value from DSK on the GUI in Lab demo 1.
- Training the features to obtain SVR models

DSK:

- Load the pre-trained SVR models into DSK.
- Do feature extractions from the song
- Perform classification based on the SVR models and the features extracted.

10. Speed and Memory on DSK

Our project requires great usage of memory on DSK. To avoid time wasted in allocating memory, we took care to initialize all memory at start up instead of allocating memory on the fly. Some of the demanding memory usages on DSK are:

1. 30 seconds of song => 30 * 44100 (sampling rate) * 4 (sizeof(float)) = 5.292 MB
2. Code size => around 58 KB
3. Initialization for FFT => around 26.6 KB
4. Initialization for features => around 200 KB
5. Initialization for SVR model => around 30 KB

Thus, in total, we used around 6 MB of the memory.

Speed wasn't a big issue in our project since we are not targeting real time emotion classification of songs. As a result, we didn't optimize for any efficient time performance. Our demo is able to classify a song of length 60 seconds in about 8 seconds.

11. Results

Method	Arousal	Valence	Combined
Training	95%	92.5%	87.5 %
Cross validation	82.5%	72.5%	N/A
Development	86.67%	66.67%	53.33%
Testing	62.5%	68.75%	50.00%

Table 1. Final results on Nu-SVR using RBF kernel based on average labels of 3 persons

Table 1 shows the final accuracy of our system using Nu-SVR and RBF kernel. For arousal, we used four features – average fundamental frequency, standard deviation of fundamental frequency, number of pitch higher than 0.25 * max pitch, and average centroid. For valence, we used five features – standard deviation of energy, average fundamental frequency, standard deviation of fundamental frequency, average centroid, and 75 percentile of beats.

	Angry	Happy	Relaxed	Sad
Angry	3	0	0	2
Happy	0	2	1	2
Relaxed	0	0	0	1
Sad	0	1	1	3

Table 2. Confusion matrix for the 16 testing songs where the rows are the real ground truths (average arousal and valence) while the columns are the system output.

Table 2 shows the confusion matrix of the 16 testing songs. From the confusion matrix, we can see that most of the classification errors are caused by errors in labeling between the x-axis or the y-axis, such as differentiating between sad/relaxed, relaxed/happy, and sad/angry. Only three errors are caused by misclassifications across the diagonals, where all three of them are from sad/happy.

However, the numbers above might not be significant because emotion is subjective and our evaluation above is based on comparing the output of our system to the ground truths (defined before to be the average of our three annotations). We cannot be sure that the ground truths are accurate as they only include annotations from three persons. In fact, it is frequent that the annotations among us are so different that they

belong to different categories of emotion. For example, for the 4th song, Beautiful Soul by Jesse McCartney, one of the annotator labels it as happy while the other two label it as sad. As a result of taking the average, the song was "categorized" as angry by us. This suggests the flaw in our datasets and evaluation system.

Thus, we come up with multiple alternate ways of evaluating to gain more insight into our system performance. We tried instead to use each person's annotations as the ground truths and get accuracy rate per person instead.

Method	Person 1	Person 2	Person 3
Training	62.5%	60%	57.5 %
Development	26.67%	53.33%	66.67%
Testing	37.5%	50.00%	32.25%

Table 3: Results of comparing the system output with individuals' annotations

Here, we are still using the exact output in the first evaluation, which was obtained by training on the average arousal and average valence. We tried to see whether the output is closer to anyone. From the results, it seems like the system performs very consistently with regards to Person 2's annotations, while it performs really badly in the testing data for Person 1 and Person 3. This suggests that Person 1 and Person 3 might not be consistent when they annotate the songs, resulting in some human errors in annotations. Another possible reason is that Person 2 tends to annotate values close to the boundary, thus making the average skewed towards Person 2's favor.

We also tried subjective evaluations of the results, by listening to the song and looking over the annotations and subjectively decide whether the annotations are acceptable. The results are shown in Table 4. As evident from the table, the accuracy rate increases. This makes sense because there are cases where our system annotated some songs to be at some boundary between two emotions. By subjectively evaluating the output, we are more lenient and allow songs to be considered accurate as long as the classification is not too absurd.

Method	Person 1
Training	77.5%
Development	66.67%
Testing	68.75%

Table 4: Subjective evaluations of system output

Finally, instead of averaging the features over 30 seconds frame, we tried to obtain a "continuous" version of annotations by looking at all partial annotations per 30-seconds frame and check whether the emotion changes is acceptable according to the 30-seconds frame. We tried this evaluation on a few songs and didn't obtain satisfying results. For example, when we ran our system on the first song, An Angel by Declan Galbraith, our system first tagged the song as an angry song for the first 30 seconds, and only tagged it as a sad song at the second 30-seconds. However, when we listened to the real song, we believe that the song should be sad for the whole period. This suggests that our SVR model currently doesn't support continuous annotations of emotions yet. This is most possibly because when we train our SVR model, we use the features averaged across 30 seconds frame, instead of smaller and independent frames. A different way of training the SVR model might allow us to have good results on continuous annotations of song emotions.

During the final demo, we also showed that our system can detect emotions from songs that are not standardized as in Section 5. We ran our system on a few Korean songs and successfully obtain acceptable emotions that the audience agrees upon. This is exciting as it suggests that our system is language-independent and may be able to deal with songs that are not standardized.

12. Future Works

Our project is far from being done. There are still lots of improvements that are possible in our system, such as:

1. Retrain the model to support "continuous" emotion annotations of songs by using smaller frame size (instead of 30 seconds) for training.
2. Test our system on songs of different languages and see whether the system is language-independent.
3. Test our system on songs of different genre such as techno, pops, classics, and etc to see whether our system is genre-independent.
4. Extract more features from the song such as tempo, harmonics, key, rhythm and etc to hopefully capture the emotion more accurately.
5. Investigate the use of other classification system such as linear regression to compare the performance.

13. Semester Schedule

Week 1	Feb 14	Background research and reading papers (everyone).
Week 2	Feb 21	Decide features and look for existing implementations (Swetha and Zhi Yang) Look up information on SVR (Chun How)
Week 3	Feb 28	Gather the song datasets (Swetha, Chun How)
Week 4	Mar 7	Standardization of songs (Chun How)
Week 5	Mar 14	Manually annotate the songs (everyone) Implement features on Matlab (Swetha, Zhi Yang) Set up LIBSVM (Chun How)
Week 6	Mar 21	Run SVM trainings (Chun How)
Week 7	Mar 28	Midterm progress presentation (everyone)
Week 8	Apr 4	Porting features extraction from Matlab to C (Swetha, Zhi Yang) Run SVR trainings (Chun How)
Week 9	Apr 11	Get feature extractions to work on DSK (Chun How) Implementing lab demo 1 (Zhi Yang)
Week 10	Apr 18	Implementing lab demo 2 (Chun How)
Week 11	Apr 25	Final presentation and demo (everyone)
Week 12	May 2	Final report (everyone)

References:

[1] B. Huang, S. Rho, R. Dannenberg, and E. Hwang. "SMERS: Music Emotion Recognition Using Support Vector Regression". *10th International Society for Music Information Retrieval Conference (ISMIR 2009).* 2009.

[2] MP3 Gain. http://www.cnet.com/1990-7899_1-6350954-1.html. Accessed on March 31, 2011.

[3] Dan Ellis. mp3read and mp3write for Matlab. http://labrosa.ee.columbia.edu/matlab/mp3read.html. Accessed on March 15, 2011.

[4] Emmanouil Benetos. Feature Extraction Tools for Audio. *Sound Description Toolbox* http://www.ifs.tuwien.ac.at/mir/muscle/del/audio_tools.html#SoundDescrToolbox

[5] Beat Detection http://archive.gamedev.net/reference/programming/features/beatdetection/

[6] Smola, Alex J., et.al. "A Tutorial on Support Vector Regression". *Statistics and Computing*, Vol.14, pp.199-222, 2004.

[7] C. Chang and C. Lin. "LIBSVM – A Library for Support Vector Machines". http://www.csie.ntu.edu.tw/~cjlin/libsvm/. Accessed at Feb 13, 2011.

[8] Kevin Wayne. How To Play an MP3 File in Java. http://introcs.cs.princeton.edu/faq/mp3/mp3.html. Accessed at Apr 17, 2011.

[9] David Kosbie. JComponentWithEvents.class. http://www.kosbie.net/cmu/spring-09/15-100/. Accessed at Apr 3, 2009.

Appendix 1: Songs list.

Here is the list of songs in our datasets, along with the corresponding arousal and valence output by our SVR model.

Author/Artist	Song	Arousal	Valence	Label
Declan Galbraith	an_angel	-0.48	-0.22	Sad
Take that	back_for_good	-0.22	0.37	Relaxed
Cascada	bad_boy	0.28	0.42	Happy
Jesse McCartney	beautiful_soul	-0.25	0.02	Relaxed
Bon Jovi	billy_get_your_gun	0.29	0.10	Happy
Backstreet boys	boys_will_be_boys	-0.18	0.02	Relaxed
Audioslave	bring_em_back_alive	0.37	-0.13	Angry
Dr Bombay	calcutta	0.28	0.67	Happy
Linkin Park	crawling	0.28	-0.08	Angry
Gym Class Heroes	cupids_chokenhold	-0.08	0.02	Relaxed
Luther Vandross	dance_with_my_father	-0.50	-0.33	Sad
Backstreet boys	everybody	0.20	-0.05	Angry
Britney Spears	everytime	-0.28	-0.17	Sad
Audioslave	exploder	0.42	-0.18	Angry
Secondhand Serenade	fall_for_you	-0.43	-0.38	Sad
Josh Groban	Galileo	-0.47	0.10	Relaxed
Backstreet boys	get_down	0.22	0.08	Happy
Headlights	get_going	-0.22	0.15	Relaxed
Avril Lavigne	girl_friend	0.47	0.02	Happy
Josh Groban	hidden_away	-0.43	-0.10	Sad
Josh Groban	higher_window	-0.43	-0.07	Sad
Greenday	holiday	0.38	0.25	Happy
Audioslave	hynoptize	0.08	-0.08	Angry
Aerosmith	i_dont_want_to_miss_a_thing	0.02	-0.07	Angry
Backstreet boys	i_ll_go_anywhere_for_you	-0.12	-0.15	Sad
Nelly Furtado	i_m_like_a_bird	-0.23	0.05	Relaxed
Ronan Keating	if_tomorrow_never_comes	-0.38	-0.13	Sad
The Duke and the King	if_you_ever_get_famous	-0.50	0.02	Relaxed
Bon Jovi	just_older	0.25	0.18	Happy
Black Eyed Peas	lets_get_it_started	0.23	0.05	Happy
Rob Thomas	little_wonders	-0.18	0.42	Relaxed
Taylor Swift	love_story	0.13	0.13	Happy
Ray Greene	my_best_friends	0.10	0.57	Happy
Celine Dion	my_heart_will_go_on	-0.50	-0.22	Sad
Westlife	my_love	0.13	-0.18	Angry

S Club 7	never_had_a_dream _come_true	-0.43	-0.20	Sad
Boyzone	no_matter_what	-0.08	-0.32	Sad
Bosson	one_in_a_million	0.03	-0.20	Angry
Blue	one_love	-0.32	0.25	Relaxed
Linkin Park	papercut	0.47	-0.05	Angry
M2M	pretty_boy	-0.38	0.03	Relaxed
Backstreet boys	quit_playing_games_ with_my_heart	0.32	-0.13	Angry
Bon Jovi	runaway	0.38	-0.32	Angry
Westlife	seasons_in_the_sun	-0.03	0.12	Relaxed
Audioslave	set_it_off	0.50	-0.10	Angry
Audioslave	shadow_in_the_sun	0.10	-0.33	Angry
Audioslave	show_me_how_to_live	0.47	-0.12	Angry
Backstreet boys	show_me_the_meaning_ of_being_lonely	-0.12	-0.18	Sad
Eminem	smack_that	0.12	-0.32	Angry
Micheal Bubble	Sway	-0.13	0.32	Relaxed
Jason Mraz	the_beauty_in_ugly	-0.05	0.02	Relaxed
B-52's	the_chosen_one	0.17	0.28	Happy
M2M	the_day_you_went_away	-0.32	0.18	Relaxed
Europe	the_final_countdown	0.25	0.22	Happy
Giorgio Vanni, Cristina D'Avena	the_johto_league	0.28	0.02	Happy
Audioslave	the_last_remaining_light	-0.25	-0.48	Sad
Elizabeth Cook	times_are_tough_in_ rock_n_roll	-0.03	0.53	Relaxed
Enrique Iglesias	tired_of_being_sorry	0.03	0.13	Happy
Bon Jovi	to_the_fire	0.25	-0.35	Angry
Hope Sandoval	Trouble	-0.33	-0.33	Sad
Bon Jovi	u_give_love_a_bad_name	0.27	-0.30	Angry
Natasha Bedingfield	unwritten	0.08	0.18	Happy
Nick Drake	way_to_blue	-0.45	-0.37	Sad
Quuen	we_are_the_champions	0.18	-0.02	Angry
Backstreet boys	we_ve_got_it_going	0.13	0.48	Happy
Linkin Park	what_i_ve_done	0.13	-0.25	Angry
Audioslave	what_you_are	0.10	-0.05	Angry
Westlife	when_you_tell_me_that _you_love_me	-0.38	-0.22	Sad
Fort Minor	where_d_you_go	0.07	0.22	Happy
Dido	white_flag	-0.30	-0.27	Sad
Blue	you_make_me_wanna	0.20	0.50	Happy