MARSYAS SUBMISSIONS TO MIREX 2012

George Tzanetakis
University of Victoria
Computer Science
gtzan@cs.uvic.ca

ABSTRACT

Marsyas is an open source software framework for audio analysis, synthesis and retrieval with specific emphasis on Music Information Retrieval. It is developed by an international team of programmers and researchers led by George Tzanetakis. In MIREX 2012 the Marsyas team participated in the following tasks that we have participated in the past: Audio Classical Composer Identification, Audio Genre Classification (Latin and Mixed), Audio Music Mood Classification, Audio Music Similarity and Retrieval, and Audio Tagging Tasks. In addition we participated for the first time with baseline systems to audio key detection and multiple F0 estimation and tracking. Finally, the INESC Beat Tracker (IBT) that is written in Marsyas participated in the beat tracking task but it is described in a separate abstract. In this abstract we describe the specific algorithmic details of our submissions and provide information about how researchers can use our system using the MIREX input/output conventions on their own datasets. Also some comments on the results are provided.

1 INTRODUCTION

Marsyas is an open source software framework for audio processing with specific emphasis on Music Information Retrieval (MIR). It has been around since 1999 and in 2002-2003 underwent a major restructure/rewrite (version 0.2) [8]. This version has now matured and has been progressing nicely in 2006-2012 with the addition of several new developers and finally some decent documentation. We have participated in several tasks mostly related to classification and similarity since the Music Information Retrieval Evaluation Exchange (MIREX) in 2007. Since 2009 we also submitted algorithms for automatic onset detection, beat tracking, and automatic music tag annotation. This year we did not submit an algorithm to the automatic onset detection.

There are two main advantages of *Marsyas* compared to other solutions for building MIR systems:

- **Integration:**

  Marsyas strives to support all the necessary algorithmic and software building blocks required to build full MIR systems. Frequently MIR researchers use a variety of different software systems to achieve their goal. For example MATLAB might be used for feature extraction and WEKA might be used for machine learning/classification. There are two main problems with such non-integrated approaches. The first is reduced performance due to communication bottlenecks between each part of the process. The second which is more deep but not really utilized in our submissions this year is the ability of integrated systems to combine signal processing and machine learning on several different abstraction layers and with both bottom-up and top-down processing. In constrast typically the use of non-integrated approaches follows the classic bottom-up sequential approach of feature extraction followed by classification.

- **Runtime performance:**

  As most practitioners of MIR for audio signals know, it takes a lot of computation time. One of the major goals of Marsyas is to reduce this computation time as much as possible. Unlike many other computer applications, computation time differences in audio MIR can play an important role in the ability to conduct experiments especially over large audio collections. An experiment that completes in 30 minutes is much easier to handle compared to one that completes in 8 hours. Fast computations means that the experiment can be repeated several times to tune different parameters. Being able to process a million sound clips can result in better statistics for feature extraction than processing 100 sound clips and so on. Marsyas achieves fast run-time performance using a variety of different means which include: 1) a dataflow architecture that minimizes the need for memory allocation and can process audio files using large networks of computation blocks with a fixed memory footprint 2) fast, optimized C++ code for all operations 3) the ability to process large collections of audio files in one run with fixed memory footprint. Frequently other approaches to MIR operate on one file at a time adding significant computation time to start/stop a process, allocate memory etc every time a file is processed.

The main goal of our MIREX submission was to highlight these characteristics of *Marsyas* and hopefully motivate more researchers to explore the framework and contribute to it. Anyone can download the software framework, look at the corresponding code and run experiments on their own datasets. In fact the source distribution of Marsyas includes a subdirectory named MIREX with specific detailed instructions of how to compile and run the MIREX tasks so that researchers can easily perform their own experiments on datasets as long as they follow they MIREX conventions. The subversion revision numbers for each year are also provided so that results from previous years can be replicated.

For the classification, tag and similarity retrieval tasks the selected set of features and classification approach we choose to utilize was straight-forward, well-known and most importantly fast to compute. The features were mostly related to timbral information. Moreover, we have significant experience using these features over a large number of various audio datasets so we felt more confident about their robustness dealing with unknown audio collections. This year we have also added features based on pitch but removed features related to rhythmic content as in our experiments their effect was minimal and they were slower to compute. More complicated feature extractors for example based on rhythmic, pitch, and stereo information are supported at various levels of completeness in *Marsyas* but unfortunately will have to wait for another MIREX.

2 TEAM

George Tzanetakis is the author of the abstract but several Marsyas developers participated in the development of the algorithms. Steven Ness (University of Victoria, Canada), Anthony Theocharis (University of Victoria, Canada) and Luis Gustavo Martins (Catolica University, Portugal) worked on various aspects of the automatic tag annotation submission. Fabien Gouyon (INESC Porto, Portugal), Joao Lobato Oliveira (INESC Porto, Portgual) and Luis Gustavo Martins (Catolica University, Portugal) worked on automatic beat detection while Luis Gustavo Martins worked on the audio onset detection. It is possible that there are other submissions by different temas that also utilized Marsyas. For example Renato Panda participated in the various classification tasks using features partially computed using Marsyas. This report only describes the submissions coordinated by the main Marsyas team.

3 SYSTEM DESCRIPTION

For all the classification, annotation and similarity tasks we participated we decided to represent each audio clip as a single feature vector. Even though much more elaborate audio clip representations have been proposed in the literature we like the simplicity of machine learning and similarity calculation using single feature vectors per audio clip. Coupled with a decent classifier this approach worked reasonably well compared to other much more complex ones.

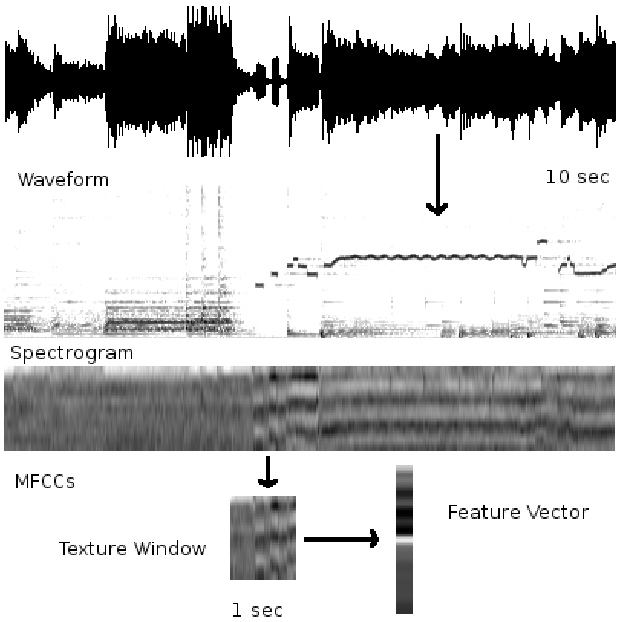

Figure 1. *Feature extraction and texture window*

The features used are Spectral Centroid, Rolloff, Flux and Mel-Frequency Cepstral Coefficients (MFCC). To capture the feature we compute a running mean and standard deviation over the past M frames:

$$m\Phi(t) = mean[\Phi(t-M+1), .., \Phi(t)] \quad (1)$$

$$s\Phi(t) = std[\Phi(t-M+1), .., \Phi(t)] \quad (2)$$

where $\Phi(t)$ is the original feature vector. Notice that the dynamics features are computed at the same rate as the original feature vector but depend on the past M frames (40 in our case corresponding to approximately a so called “texture window” of 1 second). This results in a feature vector of 32 dimensions at the same rate as the original 16-dimensional one. This process is illustrated in Figure 1. The sequence of feature vectors is collapsed into a single feature vector representing the entire audio clip by taking again the mean and standard deviation across the 30 seconds (the sequence of dynamics features) resulting in the final 64-dimensional feature vector per audio clip. A more detailed description of the features can be found in Tzanetakis and Cook [7].

In addition to these features this year we also included features based on pitch content as well as rhythmic information. The pitch features are based on computing a chroma vector every 20 milliseconds. The code is based on the MATLAB code for chroma calculation provided by Dan Ellis. The ratio of the peak corresponding to each pitch class to the maximum pitch class is calculated as a features. In addition the maximum peak as well as the average value of the chroma vector are also used as features. Means and variances over a “texture window” as well well as across the entire song are calculated similarly to the timbral features.

For all the classification tasks (audio classical composer identification, audio genre classification, audio music mood classification) a linear support vector machine classifier was used.

4 AUDIO TAG CLASSIFICATION

Audio tag annotation can viewed as a problem of multi-label classification [6]. More details about our approach to this problem can be found in a ACM Multimedia paper [3] as well as a more recent ICMLA publication [5]. We use the term stacking for the classifier architecture we have adopted. Our approach is to use a distribution classifier (a linear SVM with probabilistic outputs) that can output a distribution of affinities (or probabilities) for each tag. This affinity vector can either be used directly for indexing and retrieval, or thresholded to obtain a binary vector with predicted tag associations for the particular track. The resulting affinity vector is fed into a second stage SVM classifier in order to better capture the relations between tags. This approach is a specialized case of stacking generalization [9], a method for the combination of multiple classifiers. Similar ideas have appeared in the literature under other terms such as anchor-based classification, and semantic space retrieval, but not necessarily in a multi-label tag annotation context. The general idea is to map the content-based features to a more semantically meaningful space, frequently utilizing external information such as web resources. Stacked generalization has been used for discriminative methods for multi-label classification in text retrieval [2] but using a vector of binary predictions for each label to model dependencies between them. The most closely relevant work is applied in improving multi-label analysis of music titles again using a second stage classifier on the binary predictions of the first stage classifiers which the authors term the correction approach [4].

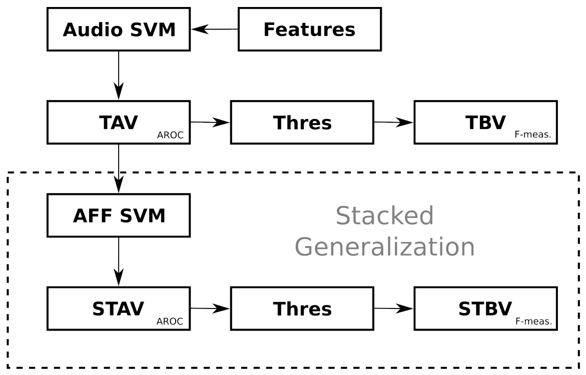

Figure 2. System flow diagram

Figure 4 shows the flow of information for our proposed audio annotation system. For each track in the audio collection a feature vector is calculated based on the audio content. As each track might be annotated by multiple tags the feature vector is fed into the multi-class Audio SVM several times with different tags. Once all tracks have been processed, the linear SVM is trained and a tag affinity output vector (TAV) is calculated. The TAV can be used directly for retrieval and storage or converted to a tag binary vector (TBV) by some thresholding method. When stacked generalization is used, the tag affinity vector (TAV) is used as a semantic feature vector for a second round of train- ing over the tracks using an affinity SVM which produces a stacked tag affinity vector (STAV) and a stacked tag bi- nary vector (STBV). The resulting predicted affinity and binary vector can be used to evaluate the effectiveness of the retrieval system using metrics such as Area under Receiver Operating Characteristic Curve (AROC) for the TAV and information retrieval measures for the TBV.

5 AUDIO MUSIC SIMILARITY AND RETRIEVAL

For the audio similarity and retrieval task once all the feature vectors (one per audio clip) have been computed the features are normalized so that the minimum of each feature is 0 and the maximum in 1 (Max/Min Normalization) and Euclidean distance over the normalized features is used for the distance matrix.

6 IMPLEMENTATION

In this section we provide information about how to download *Marsyas* and find information for installing and using the framework as well as specific information for running the tasks we participated using the MIREX 2009 input/output conventions. We hope that providing this information will help other researchers and practitioners run our system on their own datasets and motivate them to participate in the *Marsyas* developer and user communities. *Marsyas* can be compiled under Linux, OS X (Intel and PPC), and Windows (Visual Studio, Visual Studio Express, Cygwin and MinGW).

To download *Marsyas* use the following url:
`http://www.sourceforge.net/projects/marsyas`

For information and documentation use the following url:
`http://marsyas.sourceforge.net`

System specific installation instructions are provided in the documentation. Once compiled it is straightforward to run the MIREX 2009 tasks we participated. The directory MIREX in the Marsyas source tree contains all the necessary instructions including the exact SVN revision numbers that were used for the MIREX 2009 and MIREX 2010, and 2012 submissions.

For classification the *Marsyas* MIREX submissions utilized a linear support vector machine trained using libsvm [1] which is directly integrated into the source code. Figure 3 shows the *Marsyas* dataflow diagram for the feature extraction that is common among all tasks.

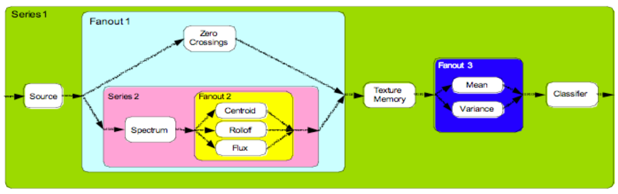

Figure 3. *Feature extraction data flow network in Marsyas 0.2*

6.1 Quick Instructions for compiling Marsyas

Quick instructions for compiling Marsyas (more detailed instructions can be found in the manual which is online at http://marsyas.info - the instructions assume that subversion and cmake are available in the system the revision number is provided separately for each task). The last command enters the subdirectory where all the Marsyas executables reside.

```
> svn -r REVNUM co SVNPATH marsyas
> cd marsyas
> mkdir build
> cd build
> ccmake ../src
> make
> cd build/bin
```

where each task has a different revision number (REVNUM). Typically the latest revision should work for all tasks however to ensure accurate replication we record the revision used for the MIREX submission. All the MIREX task instructions are availabe in the MIREX folder of the Marsyas source code repository. For example README_2012.txt is the set of instructions to run the Marsyas submissions for 2012.

7 DISCUSSION OF RESULTS

Overall we were pleased with the performance of the *Marsyas* submissions to MIREX 2012. In all tasks the Marsyas submissions performed ok. Our submissions have not changed since 2009 and are beginning to show signs of their age. For next year we plan to use a more comprehensive set of features. The detailed results are available from the MIREX 2012 webpage [1] so in this section we only briefly highlight some of the evaluation results that are specific to Marsyas. The run-time results are only available for some tasks so we can not comment in detail about the superior run-time performance of Marsyas. We expect that the Marsyas submissions are typically significantly faster especially for the classification and similarity tasks.

Finally we would like to encourage other practitioners to explore and hopefully contribute to *Marsyas*. We are also happy to offer assistance to anyone interested in porting their existing systems into *Marsyas*.

[1] `http://www.music-ir.org/mirex2012/index.php/Main_Page`

	Rank	Marsyas	Best
Composer	15/15	46.86	68.65
Latin	13/15	59.99	77.04
Mixed	12/16	67.97	76.13
Mood	14/20	60.33	67.83

Table 1. Rank and classification accuracy for Marsyas submissions

Rank	Marsyas	Best
8/10	44.87	53.188
8/10	0.894	1.138

Table 3. Fine and coarse score results of audio music similarity

8 NEW SUBMISSIONS

Both new submissions audio key detection and multiple F0 estimation and tracking performed really poorly. They were both intended to be baseline submissions that need to be improved. The audio key detection was simply done on large windows by correlating chroma vectors with the Krumhansl templates. The multiple F0 estimation and tracking was based on factorization methods but we suspect utilized a low threshold resulting in oversegmentation. Results can be found on the MIREX results webpage.

9 FUTURE WORK

There is plenty of interesting future work to be explored. Now that we have the MIREX Input/Output conventions fully supported we are very excited about participating in MIREX in the future. Our submissions this year can be considered a baseline and we can only improve in the future. In no particular order here are some of the directions we would like to explore for the tasks we participated this year: more complex audio clip representations and similarities than the single vector approach, additional features (rhythm-based, pitch/chroma based, stereo panning), and better utilization of domain knowledge such as hierarchies. In addition we hope to participate in more tasks in the following years.

10 REFERENCES

[1] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[2] Shantanu Godbole and Sunita Sarawagi. Discriminative methods for multi-labeled classification. In *Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, 2004.

[3] S. Ness, A. Theocharis, G. Martins, L., and G. Tzanetakis. Improving automatic music tag annotation us-

	MusicMiner2012	Mood2012	MusicMiner2010	Mood 2010
Marsyas F-Measure	0.4682 (9/9)	0.3745 (5/9)	0.4567	0.3672
Best F-Measure	0.4950	0.4915	0.4784	0.4658
Marsyas ROC	0.8873 (2/9)	0.8620 (2/9)	0.8828	0.8587
Best ROC	0.8917	0.8653	0.8828	0.8606

Table 2. Average Tag F-Measure and ROC for tag annotation tasks comparing 2012 and 2010

ing stacked generalization of probabilistic svm outputs. In *Proc. ACM Multimedia*, 2009.

[4] F. Pachet and P. Roy. Improving multilabel analysis of music titles: A large-scale validation of the correction approach. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(2):335–343, 2009.

[5] A. Theocharis, M. Pierce, and G. Tzanetakis. An empirical investigation of stacking for music tag annotation. In *Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on*, volume 1, pages 90–95. IEEE, 2011.

[6] G. Tsoumakas and I. Katakis. Multi label classification: An overview. *Int. Journal of Data Warehouse and Mining*, 3(3):1–13, 2007.

[7] G. Tzanetakis and P. Cook. Musical Genre Classification of Audio Signals. *IEEE Trans. on Speech and Audio Processing*, 10(5), July 2002.

[8] George Tzanetakis. Marsyas-0.2: a case study in implementing music information retrieval systems. In *Intelligent Music Information Systems*. IGI Global, 2007. to appear.

[9] David H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.