Campus Location Recognition using Audio Signals

James Sun,Reid Westwood
SUNetID:jsun2015,rwestwoo
Email: jsun2015@stanford.edu, rwestwoo@stanford.edu

I. Introduction

People use sound both consciously and unconsciously to understand their surroundings. As we spend more time in a setting, whether in our car or our favorite cafe, we gain a sense of the soundscape - the aggregate acoustic characteristics in the environment. Our project aims to test whether the acoustic environment in different areas of Stanford campus are distinct enough for a machine learning algorithm to localize a user based on the audio alone.

We limit our localization efforts to seven distinct regions on Stanford campus as enumerated in Section III-C. We characterize the locations as "regions" because we hope to capture qualitative rather than quantitative descriptions. For example, the "Huang" region includes the outdoor patio area as well as the lawn beside the building. Furthermore, we restrict our efforts to daytime hours due to the significant soundscape differences between daytime and nighttime.

A significant advantage of audio localization is the qualitative characterization on which we focus. Specifically, an acoustic environment does not generally linearly vary with position. For example, any point within a large room will likely have common acoustic characteristics. However, we expect a drastic soundscape change just outside the door or in another room, and that difference can be of significant value. However, GPS may not capture this change for two reasons:

1) This change may be below current GPS accuracy thresholds, typically 10-50 feet.
2) GPS only produces lat-long data. An additional layer of information is needed to provide information about the precise boundaries of the building.

Furthermore, GPS fails to distinguish accurate vertical position (e.g. floors), which may be of special interest in buildings such as malls or department stores.

II. Related Work

A previous CS229 course project identified landmarks based on visual features [1]. [2] gives a classifier that can distinguish between multiple types of audio such as speech and nature. [3] investigates the use of audio features to perform robotic scene recognition. [4] integrated Mel-frequency cepstral coefficients (MFCCs) with Matching Pursuit (MP) signal representation coefficients to recognize environmental sound. [5] uses Support Vector Machines (SVMs) with audio features to classify different types of audio.

III. System Design

A. Hardware and Software

The system hardware consists of an Android phone and a PC. The Android phone runs the Android 6.0 Operating system and uses the `HI-Q MP3 REC (FREE)` application to record audio. The PC uses Python with the following open-source libraries:

- Scipy
- Numpy
- statsmodels
- scikits.talkbox
- sklearn

The system also makes use of a few custom libraries developed specifically for this project.

B. Signal Flow

An audio input goes through our system in the manner below:

1) The audio signal is recorded by the Android phone
2) The Android phone encodes the signal as a Wav file
3) The Wav file enters the Python pipeline as a `Sample` instance
4) A trained `Classifier` instance receives the `Sample`
   a) The `Sample` is broken down into subsamples of 1 second in length
   b) A prediction is made on each subsample
   c) The most frequent subsample prediction is output as the overall prediction.

A graphical illustration of this is shown in Figure 1:
We have designed the system with this subsample structure so that any audio signal with length greater than 1 second can be an input.

C. Locations

The system is trained to recognize the following 7 locations:

1. Rains Graduate Housing
2. Circle of Death
   Intersection of Escondido and Lasuen

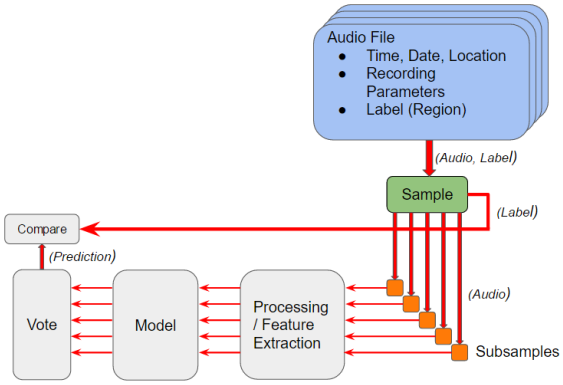

Fig. 1: System Block Diagram

3. Tressider Memorial Union
4. Huang Lawn
5. Bytes Café
6. The Oval
7. Arrillaga Gym

These locations were chosen for their geographical diversity, as well as the variety of environments. Locations 3,5, and 7 are indoors whereas Locations 1,2,4, and 6 are outdoors.

IV. Data Collection

A. Audio Format

We collected data using a freely available Android Application as noted in Section III-A. Monophonic Audio was recorded without preprocessing and postprocessing at a sample rate of 44.1 kHz.

B. Data Collection

Data was collected on 7 different days over the course of 2 weeks. Each data collection event followed the following procedure:

1) Hold the Android recording device away from body with no obstructions of the microphone
2) Stand in a single location throughout the recording
3) Record for 1 minute
4) Restart if recording interferes with the environment in some way (e.g., causing a bicycle crash)
5) Split recording into 10-second-long samples

In total, we gathered 252 recordings of 1 minute in length, for a total of 1507 data samples of 10 seconds in length. Even though our system is designed to handle any inputs of length greater than 1 second, we standardized our inputs to be 10 seconds for convenience.

We also attempted to maintain sample balance amongst the 7 locations while also diversifying sample collection temporally. The distribution of samples by location is in Table I. The distribution by day and time is given in Figure 2.

TABLE I: # Samples Gathered at each Location

Rains	Circle	Tressider	Huang	Bytes	Oval	Arrillaga
234	210	211	222	222	192	216

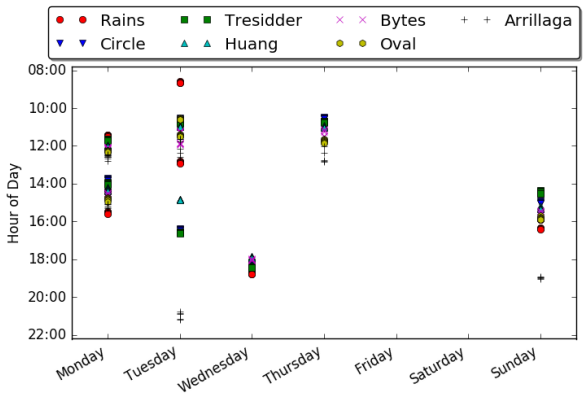

Fig. 2: Sample Distribution by Day

V. Audio Features

We investigated the use of the following features:

- Mean Amplitude in Time Domain
- Variance of Amplitude in Time Domain
- Fourier Transform (40 bins)
- Autocorrelation Function (40 bins)
- SPD (60 bins)
- 13 Mel-frequency cepstral coefficients (MFCCs)

We observed best performance using MFCC and SPD features for a total of 73 features. These 2 feature types are described in the subsequent subsections.

A. MFCC

MFCCs are commonly used to characterize structured audio such as speech and music in the frequency domain, often as an alternative to the Fourier Transform [3]–[6]. Calculating the MFCCs proceeds in the following manner [7]:

1) Divide the signal into overlapping windows
2) For each windowed signal:
   a) Take the Fast Fourier Transform (FFT)
   b) Map powers of the FFT onto the Mel scale (which emphasizes lower frequencies)
   c) Take the logarithm of the resultant mapping
   d) Take the discrete cosine transform (DCT)
   e) Output a subset of the resulting DCT amplitudes as the MFCCs

We used 23.2 ms windows and kept the first 13 MFCCs as is standard [4]. This creates multiple sets of MFCCs per signal (one per window). To summarize all of these coefficients, we take the mean over all windows of a signal. Figure 3 shows two example sets of MFCCs that obtained from different locations.

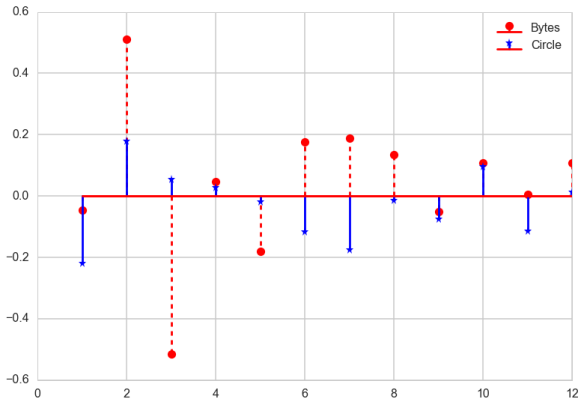

Fig. 3: Sample MFCCs at Bytes and the Circle

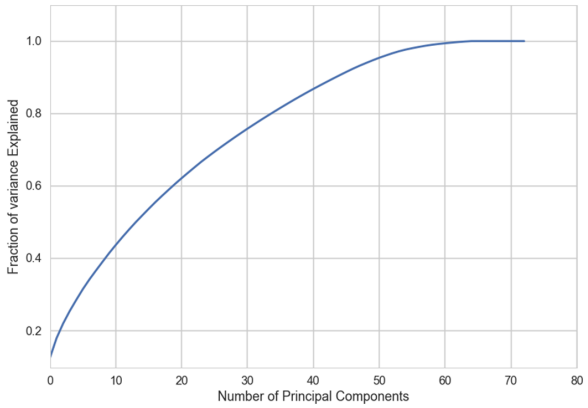

Fig. 5: Variance Explained Vs # of Principal Components

B. Spectrogram Peak Detection (SPD)

SPD is a method we developed for finding consistent sources of spectral energy over time. First, SPD generates a spectrogram using short-period FFTs, obtaining the energy of the signal as a function of both time and frequency. The method then finds the local maxima in frequency as defined by a window size.

A local maximum is marked '1', and all other elements are zero. Finally, this matrix is summed across time to give a histogram of local maxima as a function of frequency. Finally the method bins the results according to a log scale.

SPD finds low Signal to Noise Ratio (SNR) energy sources that produce a coherent signal, e.g., a motor or fan producing a quiet but consistent sum of tones. Since all maxima are weighted equally, SPD attempts to expose all consistent frequencies regardless of their power. We show a comparison of SPD outputs between the Circle and Bytes in Figure 4.

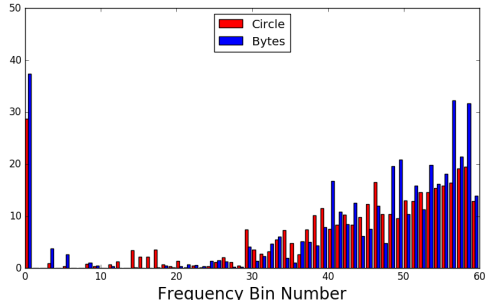

Fig. 4: Sample SPDs at Bytes and the Circle

C. Principal Component Analysis (PCA)

We investigated the redundancy in our features by doing a PCA on our data set using the above features. Figure 5 plots the fraction of variance explained vs the number of principal components used. We saw that the curve is not steep, and 50 of our 73 features probably do in fact encode significant information.

We also projected our samples onto the basis defined by the first 3 principal components for visualization. Certain regions were clearly separablein this basis, such as in Figure 6. Other regions were not quite so obviously separable, as shown in Figure 7

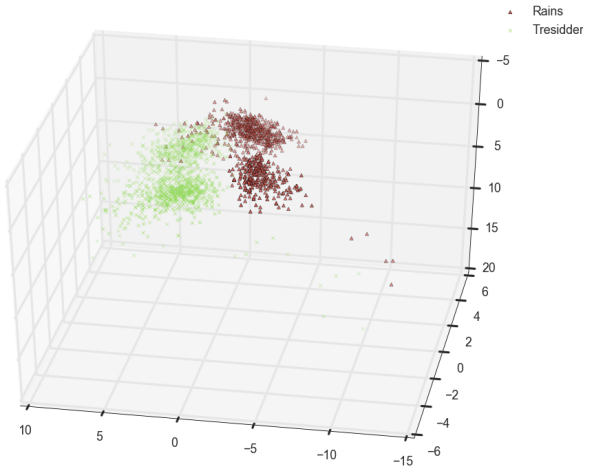

Fig. 6: Rains vs Tressider using the first 3 PCs

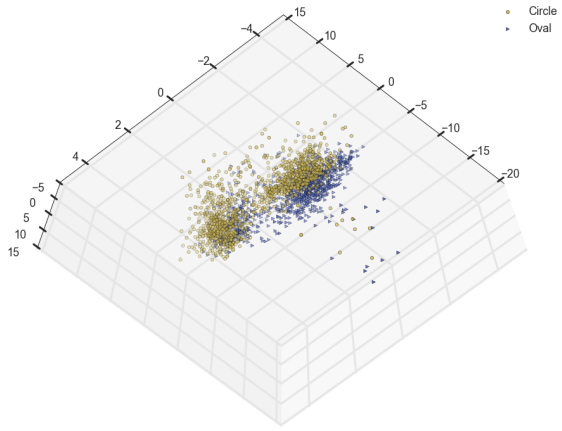

Fig. 7: Oval vs Circle using the first 3 PCs

VI. Methods and Results

Using the MFCC and SPD features, we investigated the following classifiers:

- SVM using Gaussian and Linear Kernels
- Logistic Regression
- Random Forest
- Gaussian Kernel SVM with Logistic Ensemble
  Described in more detail in the next section

When picking the hyperparameters to use for each classifier, we did a 70%-30% split of our training dataset and then searched over a grid of parameters, evaluating based on accuracy of classification.

For Logistic Regression and SVM, we also compared the use of one-vs-one (OVO) and one-vs-rest (OVR) multiclassification schemes. We found no significant difference in performance for Logistic Regression and Linear SVM. However, OVR Gaussian SVM exhibited much worse performance than OVO Gaussian SVM.

A. Voting

As described in Section III-B, our prediction method offers the following advantage: a test sample (with single label) is made up of multiple subsamples, each of which is processed and classified. The final prediction for the sample is made on a basis of majority vote from each subsample, which significantly reduces our test error. Our original implementation broke voting ties randomly. When analyzing the predictions of the Gaussian Kernel SVM, we noticed that 27% of misclassifications resulted from incorrect tie-breaks, and 42.5% of misclassifications occurred with voting margins of at most 1. We investigated 2 approaches to improving performance in these scenarios.

Our first attempt used the total likelihood produced by the SVM predictions across 10 subsamples. While this approach seemed sound in theory, the small training sample size make the likelihood estimates highly inaccurate, and this approach did not change overall performance.

Our second approach was to use the Gaussian SVM+Logistic ensemble method mentioned in Section VI. Previous testing indicated that our Gaussian kernel SVM was prone to overfitting, while the linear logistic classifier tended to have a better balance between training and test error. The final method we chose was to employ the ensemble only when the voting margin for the SVM is no more than 1. For these close call scenarios, the logistic classifier calculates its predictions for all subsamples. The SVM votes are given 1.45x weight to prevent any potential future ties, and the highest total is chosen. This method provided a 2.5% generalization error reduction.

It is also interesting to note how test error varied as we changed the duration of our test sample, effectively changing the number of votes per test sample. Using our ensemble, we achieved just under 17% error with 30 second test samples (Figure 8). This audio length is likely too long for most applications, but it is noteworthy nonetheless.

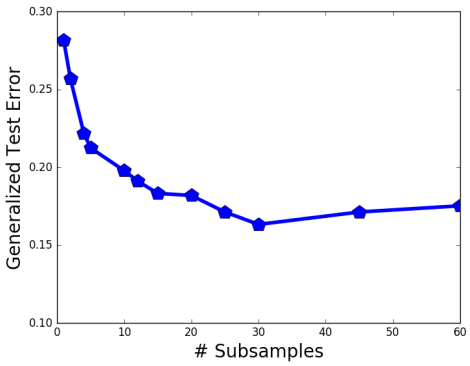

Fig. 8: Error vs. Number of Subsamples

B. Generalization

We distinguished between 2 types of testing errors:

1) Cross-Validation Error - Error on the testing set when we split the data set completely randomly
2) Generalization Error - Error on the testing set when we split based on random days.

Our data has a significant temporal correlation. We discovered that the typical Cross-Validation error was too optimistic because audio samples recorded on the same day can be significantly more correlated to each other than to audio recorded on different days. We were able to decrease our Cross-Validation error to around 8% using a Gaussian SVM. However, when we attempt to use this seemingly general classifier on a completely new day's data, we discovered it was actually very overfitted.

With this in mind, we were able to reduce our Generalization error to a bit less than 20% using a Gaussian SVM with Logistic Classifier ensemble as described in VI-A. To calculate generalization error, we did a form of 7-fold cross-validation. We held out all samples from a single day for testing while using all other days for training, and then we repeat for all 7 days during which we had gathered data. We finally do a weighted combination to calculate the Generalization Error, weighting based on the number of samples in each held out day. Table II gives a summary of our results.

TABLE II: Classifier Comparison

Classifier	**X-Validation**	**Generalization**
Gaussian Kernel SVM	13.65%	21.72%
Linear Kernel SVM	27.84%	32.74%
Logistic	15.45%	21.22%
Random Forest	14.09%	28.26%
Gaussian SVM + Logistic Ensemble	13.89%	19.68%

Using the SVM+Logistic classifier, we generated the confusion matrix in Figure 9 averaging over all hold-out trials.

Our classifier did relatively well in terms of accuracy

	Rains	Circle	Tressider	Huang	Bytes	Oval	Arrillaga
Rains	0.89744	0.05983	0.	0.01282	0.	0.02991	0.
Circle	0.14286	0.54762	0.02857	0.05714	0.02857	0.16667	0.02857
Tressider	0.	0.07619	0.8381	0.00476	0.04762	0.	0.03333
Huang	0.	0.03604	0.0045	0.91441	0.0045	0.03604	0.0045
Bytes	0.	0.01802	0.03604	0.	0.94144	0.	0.0045
Oval	0.07292	0.20833	0.01042	0.03125	0.	0.625	0.05208
Arrillaga	0.	0.04167	0.02315	0.02315	0.0463	0.04167	0.82407

Fig. 9: Overall Confusion Matrix

	Rains	Circle	Tressider	Huang	Bytes	Oval	Arrillaga
Rains	0.84091	0.06061	0.00758	0.0303	0.	0.06061	0.
Circle	0.14286	0.58571	0.01429	0.0619	0.00476	0.19048	0.
Tressider	0.00476	0.07619	0.85238	0.00476	0.02857	0.00476	0.02857
Huang	0.	0.04505	0.0045	0.90541	0.0045	0.04054	0.
Bytes	0.02193	0.02193	0.05263	0.	0.89912	0.00439	0.
Oval	0.06989	0.16667	0.01075	0.02688	0.	0.6828	0.04301
Arrillaga	0.	0.06452	0.03226	0.01613	0.05376	0.04301	0.79032

Fig. 10: Confusion Matrix with Balanced Classes

for most regions. However, the Oval and Circle are often confused for each other in a relatively balanced manner, but the Circle is frequently missclassified as Rains whereas Rains is not often mistaken for the Circle. To eliminate any effects due to our data collection's minor class imabalance (Table I), we also trained on a completely balanced data set to obtain Figure 10.

There are no major changes when balancing the dataset. This suggests that the Oval and Circle are very similar in terms of soundscape and temporal variability, a conclusion that is also supported by PCA in Figure 7. However, the Circle is likely very similar to Rains on certain days, but Rains has a more constant soundscape that is easy to identify.

C. Classifier Evaluation

As the final step in evaluating our system, we compared the performance of our classifier to people's ability to localize based on audio clips. We created a small game that would present the user with a random 10 second audio clip from our dataset. The user would then choose from which of the 7 locations the audio was taken. The pool of participants comprised of Stanford CS229 students and other attendees of our poster presentation. The results are shown in Table 11. The sample size only consisted of 41 sample points. Furthermore, we acknowledge that they did not explicitly undergo any 'training' and relied only on recall. However, it seems apparent that even Stanford students, who frequent the chosen locations, are ill-adept at identifying them by sound alone. As a baseline, random prediction would give 86% error on average with 7 labels. Of the 41 audio samples, students accurately located only 11 of them for an error rate of 73.2%. This is much higher than our classifier's generalization error of 19.68%.

VII. Future Work and Conclusion

A major challenge in this project was data collection. Due to the limited number of audio samples collected, our efforts to develop additional relevant features generally

	Rains	Circle	Tressider	Huang	Bytes	Oval	Arrillaga
Rains	0.3333	0.	0.	0.6667	0.	0.	0.
Circle	0.125	0.25	0.	0.25	0.	0.375	0.
Tressider	0.	0.	0.	0.	0.	0.	1.
Huang	0.3333	0.5	0.	0.	0.	0.1667	0.
Bytes	0.1429	0.	0.2857	0.2857	0.	0.1429	0.1429
Oval	0.2857	0.	0.	0.	0.	0.5714	0.1429
Arrillaga	0.	0.125	0.	0.	0.25	0.125	0.5

Fig. 11: Human Confusion Matrix

resulted in overfitting. Significantly increasing our training set may allow exploring additional features. In particular, we believe hour-of-day and day-of-week could be significant additions, especially to mitigate the temporal challenge of classification. As discussed in Section VI-B, we observed a gap between cross validation error and generalization error. As we utilized more data, we observed this gap lessening even with just the current set of features. We expect that our algorithm's ability to predict new data would continue to improve with additional training data. Finally, increasing our training set would make the likelihood estimates of our classifiers more accurate. Thus, it may be worthwhile to revisit the use of likelihood estimates in our voting scheme as described in Section VI-A.

The student testing we performed, as described in Section VI-C, demonstrate the challenges of audio-based localization. Users frequently noted that their 10-second clip did not seem to match the 'typical' soundscape of the area they imagine. Given the variability of soundscape at each region between different times and days, we are encouraged by our algorithm's performance. However, significant work remains to be done before conclusions can be reached about the feasibility of this method for broader applications. In particular, it is unknown how scaling the number of regions affects prediction accuracy. It would also be interesting to see our chosen features and techniques applied to very different environments with the same number of regions.

References

[1] A. Crudge, W. Thomas, and K. Zhu, "Landmark recognition using machine learning," *CS229 Project*, 2014.

[2] L. Chen, S. Gunduz, and M. T. Ozsu, "Mixed type audio classification with support vector machine," in *2006 IEEE International Conference on Multimedia and Expo*, July 2006, pp. 781–784.

[3] S. Chu, S. Narayanan, C. c. J. Kuo, and M. J. Mataric, "Where am i? scene recognition for mobile robots using audio features," in *2006 IEEE International Conference on Multimedia and Expo*, July 2006, pp. 885–888.

[4] S. Chu, S. Narayanan, and C. C. J. Kuo, "Environmental sound recognition with time and frequency audio features," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 6, pp. 1142–1158, Aug 2009.

[5] G. Guo and S. Z. Li, "Content-based audio classification and retrieval by support vector machines," *Neural Networks, IEEE Transactions on*, vol. 14, no. 1, pp. 209–215, 2003.

[6] J.-J. Aucouturier, B. Defreville, and F. Pachet, "The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music," *The Journal of the Acoustical Society of America*, vol. 122, no. 2, pp. 881–891, 2007.

[7] L. Rabiner and B.-H. Juang, "Fundamentals of speech recognition," 1993.