A Step-by-Step Approach to Footstep Detection

Francisco Cai, David Philipson, Salik Syed

December 10, 2010

1 Introduction

In this project, we tackle the problem of recognizing footsteps in audio recordings. While footstep sounds may vary from individual to individual, in general they have similar characteristics that make this problem amenable to learning algorithms. We also want to determine if it is possible to extract other information, such as the number of people walking, solely from audio recordings. There are many practical applications for footstep recognition and pedestrian count estimation, including robot-human interactions and home automation. Surveillance systems could take a step into the future by applying footstep recognition to selectively review key points in the footage.

2 Background Research

To kick-start our research, we examined literature from the fields of music information retrieval and speech recognition. The leading classification features we found were the Mel-Frequency Cepstrum Coefficients. These are primarily used for speaker recognition or simple speech recognition. The MFCC is a spectrum of a spectrum – a Fourier transform of the audio signal is then passed through a discrete cosine transform. Many of the algorithms we encountered seemed overly complex for our problem as it focused on resolving ambiguities between a large space of classifiable words using statistical or linguistic models. We felt this was not necessary for our problem – and decided a simple set of useful features would be more fruitful than complex algorithms. We also looked into literature that discussed the classification of audio events, but did not find previous research specific to footsteps.

3 Footstep Recognition

In the first part of our project, we applied machine learning to recognize the footsteps in an audio recording.

3.1 Data Collection

We did not have any reliable source of data for our learning problem. Taking this in stride, we generated our own dataset using audio recording equipment. We used a fairly sensitive studio microphone as well our laptops built-in microphones to record positive and negative samples. Positive samples were recordings of us walking around in various footwear with different background noises. Negative samples were recordings of us talking, moving around in our chairs, etc.

3.2 Implementation

We used Matlab for our project–it was easy to find and adapt existing audio processing Matlab functions for our purposes. To apply machine learning, we used Weka, a standalone Java program.[1] We first read in training examples (audio recordings in .wav format) into a Matlab matrix, calculated feature vectors from those training examples, and wrote the feature data to file using the Attribute Relation File Format required by Weka. Using Weka, we could then try using different machine learning algorithms on the data. We also used LibLinear to integrate the machine learning code with the rest of our Matlab code.[2]

3.3 Experiments

For the first pass, we chopped up each positive and negative audio recording (sample rate 44.1 kHz) into clips of 512 samples, which we then used as training examples. For our features, we used Mel-frequency cepstrum coefficients. We experimented with logistic regression, random forest classifier and sequential minimal optimization and tested using 10-fold cross-validation. The algorithms all gave us mediocre results (around 70% accuracy). We tried simple adjustments to how we were obtaining our clips - such as making them longer (as long as 64000 samples), and cutting clips at zero-crossings (periods of silence or near-silence). That gave us slight improvement in our accuracy–slightly above 80 %.

By changing how we were generating our training examples, we were able to increase our accuracy significantly, to more than 99%. In our first pass, we had been indiscriminately chopping up audio recordings containing footsteps, and labeling every clip as a footstep. This is akin to taking a photo containing mugs, cutting it out into squares, and then labeling every square as a mug. On this iteration, we used a free audio editor Audacity to manually splice out clips containing footsteps. These new clips containing only footsteps served as our new positive training examples. These clips ranged from averaged a quarter of a second in length. For negative training examples, we continues to chop the negative audio recordings into clips of similar lengths.

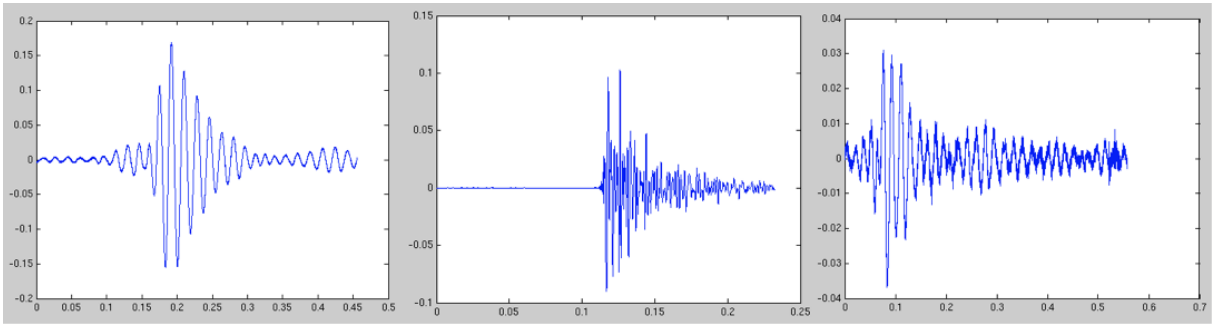

Figure 1: Waveforms of some of our training clips. From left to right: A footstep, a tap on a table, and ambient noise.

With these new training examples and using MFCC as features, we ran the SMO algorithm

[1]Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

[2]R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A Library for Large Linear Classification, Journal of Machine Learning Research 9(2008), 1871-1874. Software available at http://www.csie.ntu.edu.tw/ cjlin/liblinear

(with a linear kernel) on a training set with 255 positive examples and 1151 negative examples, with 10-fold cross validation as before. We correctly classified 99.8% of the examples, and had precision and recall above 99% for both the positive and negative examples.

Now we wanted to see how robust these results were. To test this, we decided to introduce more difficult training examples. For positive examples, that meant more noise in background–for this, we overlaid ambient noise tracks (such as ocean waves) on our footstep clips. For negative examples, we introduced more variety–computer-generated sinusoidal and noise tracks as well as noises likely to be found in a living space (e.g. hand tapping on a table). Our accuracy dropped only very slightly to 97%. Summarized results of these experiments can be found in Table 1.

	Experiment 1	Experiment 2	Experiment 3
Training Examples	Entire Recordings	Short Clips	Trickier Short Clips
Number of Examples	19160	1406	1508
Accuracy	74.53%	99.79%	97.81%
Precision(+)	0.703	1.00	0.994
Recall(+)	0.743	0.988	0.908
Precision(-)	0.783	0.997	0.974
Recall(-)	0.747	1.00	0.998

Table 1: Footstep recognition results from different experiments. (+) denotes positive (footstep) examples and (-) denotes negative examples. The first experiment used a random forest classifier, and the other two used SMO to produce a SVM model. Ten-fold cross validation was used.

The final experiment we did was to compare the performance of human versus machine on the footstep recognition task. To do this, we took a 10 second sample from one of the recordings containing footsteps and had one of the group members label the times at which the footsteps occurred. To have the machine label this 10 second sample, we first used LibLinear to train a SVM model in Matlab. Then we ran a 0.25 second sliding window over the entire sample, shifting the window over by 0.1 second each time. The start times of the windows that were classified positive with decision value above a certain threshold were then labeled as footsteps. The result of this experiment is presented in Figure 2.

4 Pedestrian Count Estimation

In the second part of the project, we applied machine learning to estimate the number of people walking in an audio recording. We built on our work from the footstep recognition part of the project.

4.1 Data Collection

As before, we collected our own data. We recorded more than 100 five-minute long audios of zero up to four people walking on our living room carpet. We had roughly 30 training examples for each category. Footwear included sandals, sneakers, and socks only. Ambient noise included talking and music playing the background.

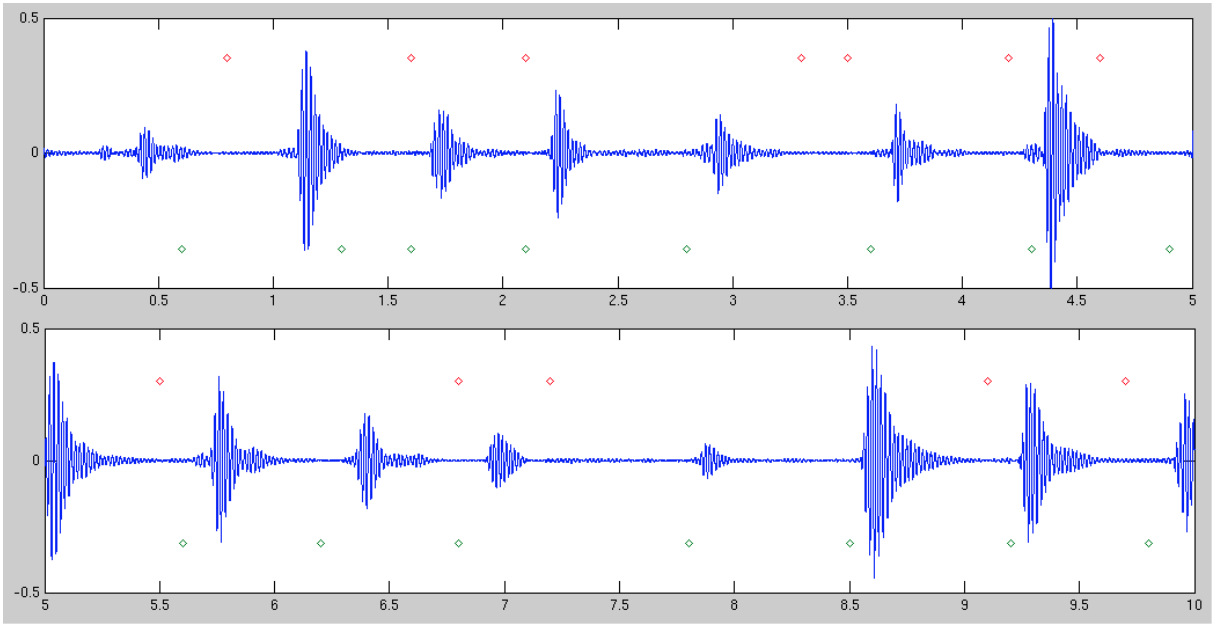

Figure 2: Human vs. machine. The waveform of the audio sample is in blue, the human labelings are the green diamonds and the machine labelings are red diamonds.

4.2 Implementation

We wrote our code in Matlab. Using LibLinear, we first trained a footstep recognition SVM using short labeled clips, as described in Section 3. Then we read in longer audio recordings (10 seconds long each) of one or more pedestrians and used our footstep recognition SVM on a sliding window over the recordings to produce a 'transcript' of footstep occurrences. This transcript is simply a set of times when footsteps were detected. In this fashion, we produced transcripts for audio recordings that contain one, two, three, or four pedestrians. Finally, using these transcripts and Weka, we trained a multi-class SVM that classifies audio recordings based on the number of pedestrians in the recording. We used Weka because it has a good user interface for trying different algorithms.

4.3 Experiments

We were inspired by how accurate SVM's can with character recognition by using very simplistic feature vectors. Thus, in our first approach we used an SVM and a 50-dimensional feature vector where the i^{th} feature f_i is the time at which the ith footstep occurred. If less than i footsteps were detected then f_i was set to infinity. Using this approach we were able to get 85% accuracy distinguishing between one or two people walking throughout the room. This approach produced much worse results when used to distinguish between 1, 2 and 3 people, with an overall accuracy of 66%. When analyzing the confusion matrix we saw that most of the misclassified examples were of three people walking. In such cases, many were misclassified as one person. One possible explanation is that the probability of overlapping footsteps increases with the number of people walking concurrently. Thus, our footstep recognition classifier may be more likely to fail since it was trained to recognize single footsteps. As a result, the transcripts generated for three people may contain fewer footstep times than those generate for two people.

We tested this hypothesis by running K-means clustering with 4 clusters and mapping each of the clusters to a different class in a way that maximizes accuracy. We found that the cluster centers for the three-people and one-person classes were fairly close to each other.

In an attempt to improve these results we decided to compute the MFCC's of the entire ten second recordings and use those coefficients as additional features in our classifier. Using this method our classification accuracy between one or two people went up to 88%. When we extended the SVM to three pedestrians the accuracy again dropped back to 66% . Overall, MFCC's did not improve our accuracy significantly. See Table 2 for the summarized results.

	1-2 people	1-3 people	1-4 people
Accuracy	87.8%	67.2%	66.2%
Precision (1 person)	0.800	0.571	0.469
Recall (1 person)	0.828	0.690	0.517
Precision (2 people)	0.839	0.656	0.647
Recall (2 people)	0.839	0.677	0.710
Precision (3 people)	n/a	0.435	0.480
Recall (3 people)	n/a	0.345	0.414
Precision (4 people)	n/a	n/a	0.710
Recall (4 people)	n/a	n/a	0.688

Table 2: Results with SVM for pedestrian count estimation.

5 Discussion and Conclusion

The results from the first part of the project show that it is possible to recognize footsteps with high accuracy, at least under controlled conditions. However, it is important to note the variables we did not address due to time constraints. A natural next step would be to obtain training data that incorporates a greater variety of footwear and walking surfaces. For our experiments, most of the training data consisted of clips in which we walked in our sandals or sneakers on our living room carpet. Shoes such as high-heels and dress shoes have very different and distinctive audio profiles. Walking on harder surfaces (such as cement) or more resonant surfaces (hardwood floors) would also result in different footstep sounds. Adding these variables would be an interesting starting point for future work.

The results from the second part of the project show that we can distinguish between one and two people walking very well, but not as well for more than two people. There are several approaches that could be taken to attempt to improve this result. One possibility is to attempt to create finer-grain transcripts. Much information is lost in the multiple-person case when several people step at or close to the same time. By refining our footstep classification to distinguish multiple footsteps in close succession, we would likely be able to improve the quality of the transcripts. Another possibility is to add additional information to the transcript beyond just the times of footsteps. For example, if our program could notice that the footsteps come in three distinct "styles," it might be able to guess that there are three people with different-sounding steps.

Despite the difficulties with larger numbers of people, our success at recognizing footsteps and classifying small numbers of people suggests that technology for gathering information about pedestrians by their footsteps is viable and a ripe possibility for further study.