Robust and Efficient Multiple Alignment of Unsynchronized Meeting Recordings

T. J. Tsai, *Student Member, IEEE*, and Andreas Stolcke, *Fellow, IEEE*

***Abstract*—This paper proposes a way to generate a single high-quality audio recording of a meeting using no equipment other than participants' personal devices. Each participant in the meeting uses their mobile device as a local recording node, and they begin recording whenever they arrive in an unsynchronized fashion. The main problem in generating a single summary recording is to temporally align the various audio recordings in a robust and efficient manner. We propose a way to do this using an adaptive audio fingerprint based on spectrotemporal eigenfilters, where the fingerprint design is learned on-the-fly in a totally unsupervised way to perform well on the data at hand. The adaptive fingerprints require only a few seconds of data to learn a robust design, and they require no tuning. Our method uses an iterative, greedy two-stage alignment algorithm which finds a rough alignment using indexing techniques, and then performs a more fine-grained alignment based on Hamming distance. Our proposed system achieves >99% alignment accuracy on challenging alignment scenarios extracted from the ICSI meeting corpus, and it outperforms five other well-known and state-of-the-art fingerprint designs. We conduct extensive analyses of the factors that affect the robustness of the adaptive fingerprints, and we provide a simple heuristic that can be used to adjust the fingerprint's robustness according to the amount of computation we are willing to perform.**

***Index Terms*—Audio fingerprint, adaptive, eigenfilter, alignment, meetings.**

I. Introduction

MORE and more, mobile computing devices are carried by their users at all times, including when they engage in meetings with others. As a result, it makes sense to explore an application scenario in which multiple mobile devices could be used to generate a reasonably high-quality recording of a meeting, offering a low-cost alternative to potentially expensive recording equipment or software. In this scenario, meeting participants would use their mobile phones, tablets or laptop computers as audio recording devices in an unsynchronized manner. No matter when they arrive at the meeting, participants place their mobile devices on the table in front of them and begin recording. Assume person A arrives at time $t = 0$ minutes and begins recording. Person B arrives at time $t = 2$. Person C joins remotely via skype at $t = 5$, and he too simply places his mobile phone in front of him at his remote location. Person D arrives late at time $t = 25$ minutes and begins recording. Some people leave the meeting early; others stay late. At the end of the meeting, everyone has an audio recording. We would like to take these partial, unsynchronized, overlapping audio recordings and generate a single high-quality "summary" recording of the entire meeting. This paper proposes a method for accomplishing this in an efficient and robust manner.

The main problem that needs to be addressed in this application scenario is to align the audio files with each other in time. Once the audio files are aligned in time, we can generate a summary recording by simply averaging the audio channels or using a blind beamforming approach. Note that the term "alignment" often refers to aligning text and audio (e.g. forced alignment), whereas here we are aligning audio to audio. No transcriptions are necessary for this type of alignment.

Note that explicit timestamping of recordings would not be a reliable way to align files. Clocks on mobile devices are not typically synchronized, and might not even record time at the required precision (on the order of milliseconds).

The most straightforward content-based alignment method is to use a simple cross-correlation method. While cross-correlation might work for aligning audio files with small time offsets, it would be prohibitively expensive for situations where the recordings are an hour long and the time offset might be 25 minutes, as in the example above.[1] Additionally, simple cross-correlation alone will not handle transitive relationships: if A and B overlap, and B and C overlap, but A and C do not overlap, we should still be able to align all three using the information given. One can immediately come up with several ideas to improve the efficiency of a cross-correlation approach: performing the cross-correlation on FFT data rather than time samples, using only a segment of audio to compute the correlation estimates, etc. This paper represents the development of one such line of thought taken to its logical conclusion.

This paper proposes an approach to the multiple alignment problem that is based on audio fingerprinting techniques. The primary novel contribution of this work is a method for learning a fingerprint representation on-the-fly in an unsupervised manner. This binary fingerprint representation is based on learning a set of spectrotemporal eigenfilters, and then allowing the fingerprint bits to represent whether the resulting spectrotemporal features are increasing or decreasing in time. The fact that this method works in an unsupervised fashion makes it possible to learn the fingerprint design on-the-fly, so that the fingerprint

Manuscript received June 04, 2015; revised November 30, 2015; accepted January 26, 2016. Date of publication February 08, 2016; date of current version March 23, 2016. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Hirokazu Kameoka.

T. J. Tsai is with Department of Electrical Engineering and Computer Science, University of California Berkeley, Berkeley, CA 94720 USA (e-mail: tjtsai@icsi.berkeley.edu).

A. Stolcke is with Microsoft Research. He resides in Berkeley, CA 94704 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASLP.2016.2526787

[1]We will compare the computation requirements of our approach with a simple pairwise cross-correlation method in Section VI.

representation is adapted to each alignment scenario (i.e. group of audio files to be aligned) rather than being fixed based on a global training set. One significant benefit of our approach is that, because the fingerprint design is learned on-the-fly, our system requires little or no tuning. The robustness of the fingerprints can be adjusted very easily by tuning two hyperparameters, according to the amount of computation we are willing to perform. Once the audio is represented in a binary fingerprint representation, the alignment is accomplished using an iterative, greedy two-stage alignment algorithm which performs a rough alignment using efficient indexing techniques, and then refines the alignment estimate based on Hamming distance.

The rest of the paper is organized as follows. Section II discusses background work on relevant topics. Section III explains the experimental setup, including a detailed description of the audio fingerprinting technique and alignment algorithm. Section IV shows the results of our experiments. Section V does an in-depth analysis of results. Section VI discusses several practical takeaway lessons. Section VII summarizes and concludes the work.

II. Related Work

Our approach for the alignment of multiple overlapping meeting recordings grows out of the development of audio fingerprinting techniques. We will introduce previous work in three areas: music identification, online copy detection, and alternative applications of audio fingerprinting.

Audio fingerprinting techniques were first developed in the context of music identification. In this scenario, a user, often with significant ambient noise, would like to identify a song that is playing in the background. By recording a noisy sample of the song, the user can search for a match in a database of known songs. There are several commercial applications for cell phones that offer this functionality, such as Shazam [1] and SoundHound [2]. We will briefly describe some well-known approaches.

Perhaps the most well-known approach is the Philips fingerprint proposed by Haitsma and Kalker [3]. In this approach, one first computes a spectrogram containing 33 Mel bands between 300 Hz and 2kHz. Each frame yields a single 32-bit fingerprint, where each bit conveys whether the energy difference in two adjacent frequency bands increases or decreases between two consecutive frames. Many works have extended this approach in various ways, such as weighting the fingerprint bits according to their robustness [4], [5], giving more emphasis to fingerprints that repeat consecutively and are thus more stable and robust [6], and distributing the database across a cluster in a way that is memory-efficient [7], [8].

Another well-known approach is the Shazam fingerprint proposed by Wang [9]. This approach first identifies local spectral peaks in the spectrogram. It then considers various pairings of peaks and constructs 32-bit fingerprints of the form $(f_1, f_2, \Delta t)$, where f_1 and f_2 denote the frequency of the two spectral peaks and Δt denotes the time difference between the peaks. This approach rests on the insight that the spectral peaks are the part of the signal that are most robust to additive noise, and that the onset of spectral peaks is very pronounced in musical signals. Several works extend this approach to allow for tempo changes [10], pitch shifts [11], or both [12], [13]. Other works explore a similar method of encoding the locations of maxima in wavelet coefficients [14], [15], spectral luminance values [16], and sparse spectrotemporal dictionary elements [17]. Some approaches use the location of local maxima in time, time-frequency, or time-feature space to determine when to compute a local fingerprint descriptor [18]–[21].

Many other works propose fingerprints based on manually designed features, such as modulation frequency features [22], [23], chroma [20], [21], spectral flatness [24], [25], and spectral subband moments [26], [27].

Several approaches to music identification have incorporated learning into the process. Ke et al. [28] improve upon the Philips fingerprint by treating the spectrogram as an image, considering a family of Viola-Jones face detection features [29], and then introducing a pairwise boosting algorithm to automatically select the most robust features and their corresponding thresholds. The works by Jang et al. [30] and Kim and Yoo [31] similarly consider a candidate set of features, and then use boosting to select the most robust set of features and thresholds. Burges et al. [32] propose a method based on training a linear convolutional neural network, where each layer performs an oriented PCA reduction.

Audio fingerprinting has also been explored in the context of online copy detection. In this scenario, we would like to detect when a user uploads copyrighted material to a file sharing website. Note that music identification and copyright detection both have the same basic problem formulation but different types of noise and distortion. In the music identification case, the distortions may include the room acoustics, additive noise from the environment, and the microphone characteristics. In the online copyright detection case, the distortions may include different audio compression qualities, frame dropping, equalization, and mixing with other audio tracks. The TRECVID content based copy detection task [33] provided a common platform to evaluate both video and audio copy detection, and it spurred much of the research in online audio copy detection [18], [34]–[43].

Beyond music identification and copy detection, audio fingerprinting techniques have also been applied to a variety of other applications. These include detecting repeating objects in audio streams [44]–[48], recognizing a TV channel in real-time [49], synchronizing two different versions of a movie [50], two TV audio streams [51], or a music video and a studio album track [52], and performing self-localization of multiple recording devices [53]. Of particular interest to this present work, several works have explored the synchronization of consumer videos of the same live event. Shrestha et al. [54] apply the Philips fingerprint to match pairs of video in order to synchronize several video recordings of the same live event. Kennedy and Naaman [55] likewise apply the Shazam fingerprint in a pairwise manner to synchronize videos of live concert recordings. Su et al. [56], Bryan et al. [57], and Six and Leman [58] extend the work of Kennedy and Naaman by applying additional post-processing steps such as clustering or a more refined alignment.

Our work explores a specific application which has hitherto not been studied: aligning unsychronized audio recordings of meetings, such as might be collected from participants' personal devices. This application scenario presents some unique challenges and requirements which lead to the development of novel audio fingerprinting techniques.

The primary contribution of our work is a method to learn an audio fingerprint design in an *unsupervised* manner. This allows the fingerprint representation to be adaptive to each alignment scenario (i.e. a set of meeting recordings that need to be aligned). Rather than fixing a fingerprint representation based on a separate training set, the fingerprint design is instead learned on-the-fly in a completely unsupervised fashion and adapted to perform well on the data at hand. Many fingerprint approaches are manually designed or learned in a supervised manner, but, to the best of our knowledge, this is the first entirely unsupervised adaptive fingerprinting method. Our method is based on learning a set of spectrotemporal eigenfilters, and then encoding whether or not the corresponding spectrotemporal features are increasing or decreasing in time as bits in the fingerprint. This method requires very little data to train (on the order of seconds of speech), is efficient to compute, and works without any labeled data, which allows us to tailor the fingerprint design to the particular characteristics of each alignment scenario.[2]

It is important to note that our current problem should not be confused with research work on microphone arrays. Research on microphone arrays focuses on small time lags between microphone elements in order to infer source location or inform beamforming weights. These works typically assume that the microphone elements record audio in a coordinated manner or are approximately synchronized, so that simple cross-correlation over small time lags can be used to align the recordings. Our focus here is on aligning audio files which might be offset by an arbitrary amount of time (e.g. 30 minutes), or may not be directly overlapping at all (i.e. A overlaps with B, B overlaps with C, but A and C do not overlap).

III. Experimental Setup

The experimental setup will be described in five parts: the fingerprint computation, the fingerprint design, the alignment algorithm, the data, and the evaluation metrics.

A. Fingerprint Computation

The fingerprint computation consists of 6 steps, which are described below.

1) **Compute spectrogram.** We used 100 ms windows in time with 10 ms hop size to generate a linear spectrogram. We then integrated over 33 Mel frequency bands between 200Hz and 2000Hz and took the logarithm of band energies. These settings are similar to those used in several previous audio fingerprinting works [3], [28], [14], [18].
2) **Collect context frames.** When computing the fingerprint at a particular frame, we consider w frames of context. So, at each frame we are working with a vector of dimension $33w$.
3) **Apply eigenfilters.** We compute a set of N features at each frame by applying N different spectrotemporal filters. In other words, each feature is a linear combination of the log Mel spectrogram values for the current frame and surrounding context frames. Note that MFCCs are a special case of spectrotemporal filters in which the filter coefficients match the coefficients of the discrete cosine transform transform. Rather than using MFCCs, however, we use filters that capture the directions of maximum variance. We will refer to these filters as eigenfilters. We will discuss our choice of spectrotemporal filters in the next section.
4) **Compute deltas.** For each of our N features, we compute the change in the feature value over a time lag T. If the feature value at frame n is given by x_n, then the corresponding delta feature will be $\Delta_n = x_n - x_{n+T}$. In our experiments, we used a time lag of 50 frames (.5 seconds). The justification for this step and for this particular choice of T will be discussed in the next section.
5) **Apply threshold.** Each of the N delta features is compared to a threshold value of 0, which results in a binary value. These bits represent whether the N features are increasing or decreasing across the time lag T.
6) **Bit packing.** The N binary values are packed into a single 32-bit integer which represents the fingerprint value for a single frame. This compact binary representation will allow us to store fingerprints in memory efficiently and to do reverse indexing, to quickly look up fingerprint matches.

Figure 1 shows a block diagram of the fingerprint computation process. The text above the boxes shows the dimension (per frame) at each stage of the process. The text below the boxes provides additional clarifying information. We will now turn our attention to an explanation and justification of *why* we use the computation process in Figure 1.

B. Fingerprint Design

We now discuss the rationale and design of the proposed fingerprint computation. Our formulation grows out of two key principles of good fingerprint design.

Design principle 1: Informativeness. A good fingerprint should represent a maximum amount of information in as little space as possible. There are two direct consequences of this principle in the context of our fingerprint. First, the threshold for each bit should be set to the median of the underlying distribution. This threshold value ensures that the fingerprint bit will have maximum entropy and thus communicate the most information. Note that if the threshold is set to an extreme value in the tail of the distribution, the bit will always be 0 (or 1) and thus communicate no useful information. Second, the fingerprint bits should be uncorrelated. Any correlations between fingerprint bits represents inefficiency. For example, a single fingerprint bit that is simply replicated 32 times will result in

[2]This paper extends our earlier preliminary work [59]. Note that the lattice projection step in our earlier work has been abandoned, as we empirically verified that it led to less robust fingerprints.

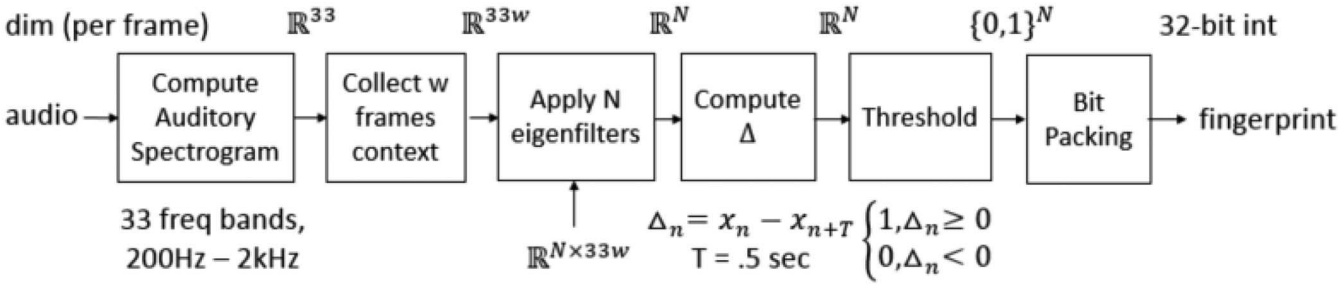

Fig. 1. Block diagram of the fingerprint computation.

a fingerprint which still only contains a maximum of 1 bit of entropy. On the other hand, a set of 32 independent bits will have the maximum 32 bits of entropy.

Design principle 2: Robustness. A good fingerprint should be robust to noise. In the context of our fingerprint design where each bit represents a feature compared to a threshold, achieving robustness corresponds to maximizing the variance of the feature distribution. To see this, note that the feature distribution will be roughly bell-shaped (as a result of the central limit theorem), and that the threshold will be set at the median of the distribution (as discussed above). If a particular feature value falls close to the threshold, a small perturbation from noise may cause the feature to fall on the other side of the threshold, resulting in an incorrect bit. This situation can be minimized by maximizing the variance of the feature distribution.

We now express these design principles in a mathematical form. Consider the n^{th} audio frame in a set of audio data, and let the log Mel spectrogram values for the w context frames be denoted $a_n \in \Re^{33w}$. Let $A \in \Re^{M\times 33w}$ denote the matrix containing all such data points a_n, where M is (approximately) the total number of audio frames in the data set. Let $x_i \in \Re^{33w}$ specify the weights of the i^{th} spectrotemporal filter, and let $S \in \Re^{33w\times 33w}$ be the covariance matrix of the data in A. Finally, let N denote the number of bits in the fingerprint. Then, for $i = 1, 2, \ldots, N$, we would like to solve

$$\begin{aligned} \text{maximize} \quad & x_i^T S x_i \\ \text{subject to} \quad & \|x_i\|_2^2 = 1 \\ & x_i^T x_j = 0,\ j = 1, \ldots, i-1. \end{aligned} \tag{1}$$

Each resulting x_i specifies the spectrotemporal filter weights for the i^{th} fingerprint bit.

Let's unpack the above formulation. The first line can be summarized as "maximize the variance." To see this, note that the variance of the features Ax_i can be expressed as $\frac{1}{M}\|\tilde{A}x_i\|_2^2$, where the columns of $\tilde{A}$ are zero mean. This objective is motivated by our second design principle (robust). The first constraint simply says, "finite energy." We could use a number of different ways to constrain the energy, and we choose the L2 norm for reasons that we will see shortly. The last constraint says, "uncorrelated filters." This constraint ensures that the filters are mutually orthogonal. This constraint is motivated by our first design principle (uncorrelated bits).

Equation (1) is exactly the eigenvalue/eigenvector problem, where x_i, $i = 1, \ldots, N$ are the N eigenvectors of S with highest eigenvalue. The benefit of this formulation is that it can be solved very efficiently using standard implementations. The eigenvectors, once "reassembled" as eigenfilters spanning w context frames, are the spectrotemporal filters that are applied in Step 3 of the fingerprint computation. These spectrotemporal filters yield the spectrotemporal features with maximum variance, ensuring that the fingerprint bits will be robust. The eigenvectors will also be orthogonal to one another, ensuring that the fingerprint bits will be uncorrelated. One big advantage of this formulation is that it can be done in an unsupervised fashion. We can thus design a robust fingerprint without labeled data.

We cannot simply threshold the spectrotemporal features themselves, however, for the resulting fingerprint would not satisfy one other important characteristic: invariance to volume level. In order to work effectively in our use-case scenario, the fingerprints must be invariant to acoustic energy level. So, the same audio signal attenuated or amplified should yield the same fingerprint values. This is important because when a person speaks, the same signal will be picked up by multiple recording nodes, but with varying attenuation levels (along with distortions, of course) depending on the distance to the speaker.

To make our fingerprint invariant to acoustic energy level, we compute delta features before applying the thresholds. If the feature at frame n is x_n, then the corresponding delta feature will be $\Delta_n = x_n - x_{n+T}$, where T represents the time lag. By symmetry, these delta features will have a distribution centered around 0, so our median thresholds will all be set to 0. Each fingerprint bit thus represents whether the features are increasing or decreasing in time (across a time lag T), which is invariant to acoustic energy level. In contrast, applying a threshold to the features themselves (rather than the delta features) would not be invariant to volume level.

Computing delta features is a more effective way to achieve volume-invariance than L2 normalization for two reasons. First, it is computationally cheaper. Computing delta features simply requires one additional subtraction per feature per frame, whereas normalizing the spectral values in a set of context windows at each frame is much more expensive. Second, the delta features are far more robust. Each delta feature can thought of as the sum of two different variables, which effectively doubles the variance of the feature and thus increases its robustness.

There is a tradeoff in the selection of the time lag T. For very small T, the delta features will have lower variance, since we are taking the difference between features that are immediately adjacent in time (and thus highly correlated). A larger T will thus yield a more robust fingerprint up until the point where the

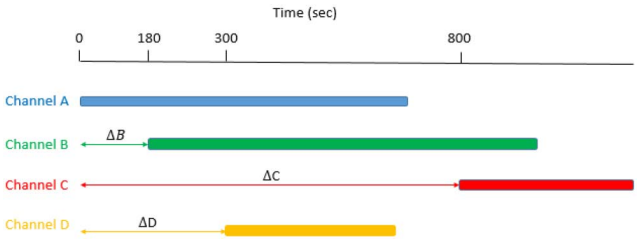

Fig. 2. Graphical depiction of the temporal alignment problem.

signal becomes decorrelated with itself. On the other hand, a very large T results in a fingerprint that is not very localized in time. So, the ideal T is the minimum time lag that ensures that the underlying audio signal has become decorrelated with itself. The selection of T could thus be determined empirically on-the-fly by measuring autocorrelation, or it could be set to a fixed value based on a priori assumptions. Given that typical speech rates in American English range between 110 - 150 words per minute, we select $T = 50$ frames (.5 seconds) as a conservative value that ensures decorrelation.

To recap, we select a set of orthogonal spectrotemporal filters which maximize the variance of the resulting feature distribution. We can do this efficiently by computing eigenvectors with highest eigenvalues. Selecting this set of eigenfilters ensures a maximally robust fingerprint. In order to make our fingerprints invariant to acoustic energy level, we insert an additional delta computation stage before applying the thresholds.

The above representation is similar to locality sensitive hashing [60], except that the data is projected onto dimensions of maximum variance, rather than projected onto randomly selected directions. In the hashing literature, this approach is known as spectral hashing [61]. So, one can think of our fingerprint computation as applying spectral hashing to auditory spectrogram values in surrounding context frames, along with a modification which ensures that the representation is volume-invariant.

C. Alignment Algorithm

In this subsection, we describe the algorithm used to align the multiple audio recordings in time. The problem of aligning multiple recordings in time is depicted graphically in Figure 2. Here, we see four different audio recordings denoted by A, B, C, and D. Person A begins recording first ($t = 0$), person B begins recording 180 seconds into the meeting, and so forth. Note the nontransitive relationships among the files: A overlaps with B, and B overlaps with C, but A and C do not overlap. It will be important for our algorithm to be able to handle these nontransitive relations, rather than simply comparing files in a pairwise manner.

The alignment algorithm has four steps, each described below.

Step 1: Initialization. The initialization step has four components. First, we determine the adaptive fingerprint design using the method outlined in the previous subsection. This determines the 32 spectrotemporal filters that are adapted to the data at hand. Second, we compute fingerprints on all the audio recordings in the current alignment scenario. In the example shown in Figure 2, this means extracting fingerprints from recordings A, B, C, and D. Third, we create a database which contains triples of the form $(fp, fileid, offset)$, where fp specifies the 32-bit fingerprint value, $fileid$ specifies the audio recording, and $offset$ specifies the frame offset relative to the beginning of the audio recording. To make the fingerprint lookups more efficient, we also create a reverse index which maps fingerprint values to the list of triples with that fingerprint value. Fourth, we select one of the audio recordings to serve as our "anchor" file. In our experiments, we selected the anchor file to be the audio recording with highest average energy. All other time indices will be computed relative to the beginning of this anchor file. We denote this time index as the universal time index. The time scale in Figure 2 shows the universal time index when recording A is selected as the anchor file.

Step 2: Find the best match. Using the anchor file as a query, we find the audio recording that has the strongest match. We use the method proposed in the Shazam algorithm [9], which is based on histograms of time differences. A brief explanation is given here, and the reader is referred to the Shazam paper for more details. For every fingerprint f at time offset $offset_{query}$ in the query file, we look up the list of matching fingerprint triples $(f, U_i, offset_{U_i})$ in the database, where U_i, $i = 1, 2, \ldots, k$ is the $fileid$ for one of the k currently unaligned audio recordings. If the query file and an unaligned file U_i overlap in time, we expect there to be a lot of matching fingerprint triples with a fixed time difference $\Delta t = offset_{query} - offset_{U_i}$. So, we can estimate the true alignment between the query and U_i by accumulating a histogram of the matching fingerprint time differences Δt, and then scanning the histogram counts for a peak. If there are a lot of matching fingerprints at a particular time offset Δt, then Δt indicates the relative alignment between the query file and the unaligned audio recording. In this way, we accumulate a histogram of time offsets for each unaligned audio recording U_i, and we take the maximum bin count of each histogram as the match score for U_i. The unaligned audio recording with the highest match score is identified as the best match.

Step 3: Fine alignment. We know the approximate offset Δt between the query file and the best match. However, this offset is not very precise, since its precision is limited by the width of the histogram bins. Also, since a fingerprint match requires all 32 fingerprint bits to be correct, the match score ignores a lot of more fine-grained information about fingerprint agreement. For these reasons, we do a fine-grained alignment between the query file Q and U^*, the unaligned audio recording with the best (rough) match score. We consider a range of possible offsets $[\Delta t - B, \Delta t + B]$, where B represents a search window size in frames. For each possible offset, we compare the corresponding fingerprints in Q and U^* and determine what percentage of the fingerprint bits agree. Note that here we are comparing individual fingerprint bits, which allows us to detect partial matches, unlike in the best match step that compares only the full 32-bit fingerprint values. These bit comparisons can be computed very efficiently using bit arithmetic, and they allow us a much more precise estimate of fingerprint agreement.

The offset Δt^* which yields the highest fingerprint agreement is selected, and it specifies the starting time of U^* on the universal time index. U^* is added to the list of aligned files.[3]

Step 4: Repeat steps 2 and 3. We repeat step 2 using the most recently aligned file as the query file. For all aligned files, frame offsets are adjusted to represent the universal time index. In other words, fingerprint tuples $(fp, fileid, offset)$ will effectively become $(fp, fileid, offset + \Delta t^*)$. When accumulating histogram counts for the current query file, we retain the histograms from previous steps and simply add additional counts. In this way, we accumulate evidence from all of the previously aligned files to help match unaligned files. This means that when we align the last recording (which will be the recording that had the lowest match scores), we will have the most data and evidence to help determine the optimal alignment. Steps 2 and 3 are thus repeated in like fashion until all files have been aligned.

At the end of this process, we have estimates of the relative alignment among all the audio recordings. Figure 2 shows a possible representation of the alignment estimates after the entire alignment process has been completed.

D. Data

To evaluate our system, we ran experiments on data extracted from the ICSI meeting corpus [62]. The original data set consists of multi-channel audio recordings of 75 real research group meetings, totaling approximately 72 hours of meetings. For each meeting, participants wore headsets which captured audio through close-talking microphones. Several tabletop microphones spread across the conference room table also collected audio data. These tabletop microphones included four high-quality omnidirectional microphones and two low-quality microphones mounted on a stand. The meetings ranged in length from 17 to 103 minutes, and the number of simultaneous audio channels ranged from 9 to 15 channels. The data set also contains manual annotations of what people said, who spoke, and when they spoke.

The ICSI meeting corpus provides useful source material that we can use to generate realistic query scenarios for the task at hand. The corpus has three important characteristics that make it suitable for our experiments. First, the data contains multiple synchronized recordings. The ICSI data set contains simultaneous audio recordings that are synchronized down to the level of milliseconds, which gives us a reliable ground truth. Second, the data has microphones placed throughout the conference room. In a realistic scenario where a group uses their portable devices to record a meeting, there will be diversity in microphone location, so this is an important characteristic to maintain. Third, the data contains a variety of microphone characteristics. In our scenario of interest, users would have different types of portable devices which would have different microphone characteristics, so diversity in microphone characteristics is an important aspect. The meeting data contains close-talking and far-field microphones, as well as both high-quality and low-quality microphones. It is useful to point out that data collected from a microphone array generally does not satisfy characteristics 2 and 3 above. While using actual data with mobile phones would be ideal, coordinating the collection of such a data set in sufficient quantity is outside the scope of this work. For the reasons described above, the ICSI meeting corpus provides good source material for our experiments.[4]

We generate each alignment scenario as follows. Given the audio data for a single meeting, we randomized the ordering of audio channels and performed the following steps on each channel in the random order.

1) Select an audio segment length from a uniform distribution $[0, T]$. In our experiments, we selected T to be 10 minutes.
2) Randomly select a time interval of this length from the full audio recording.
3) Verify that the selected audio segment has 30 seconds or more of temporal overlap with at least one other previously selected audio segment. If it does not, repeat Steps 1 and 2 until this condition is met.

In this way, each audio channel generates a randomly chosen audio segment, and every audio segment is guaranteed to have at least 30 seconds of overlap with at least one other audio segment. Since the above process is probabilistic, we can generate multiple query scenarios from a single meeting. We generated 10 query scenarios from each of the 75 meetings, resulting in a total of 750 query scenarios and approximately 8500 alignments. We used 37 of the meetings for training, and the other 38 meetings for testing. Since our fingerprint design is entirely learned on-the-fly for each alignment scenario, there is little training to do. The training set was primarily used for system debugging and for learning appropriate values for a few system parameters such as the histogram bin width.

Note that the above process of generating queries is probably more difficult and challenging than a typical use case scenario, since users would probably all record a very substantial chunk of the meeting, with an occasional user leaving the meeting early or entering very late. However, generating more difficult alignment scenarios with shorter audio segments and shorter amounts of temporal overlap will enable us to better characterize and test the robustness of our system.

E. Evaluation Metrics

We evaluate our proposed system by measuring the robustness and accuracy of the alignments in the following manner. Consider a single alignment scenario, such as the one depicted in Figure 2. If we use channel A as an anchor file, we can compute the time offset of all other files relative to A. These time offsets are denoted in Figure 2 as ΔB, ΔC, etc. Our alignment system will produce a set of hypotheses $\Delta B_{hyp}, \Delta C_{hyp}, \cdots$ for each alignment scenario. Since we generated the alignment scenario ourselves, we also know the true

[3]The method described above is useful for finding an approximate alignment on the order of the frame size (10 ms) very efficiently. If even finer precision is needed, one could do an additional cross-correlation in the time-domain around the approximate alignment.

[4]The AMI Meeting Corpus [63] would be another data set that is suitable for our study. We chose to use the ICSI meeting corpus because the meetings are not scripted, and there is greater diversity in the number of meeting participants and microphone types.

offsets $\Delta B_{ref}, \Delta C_{ref}, \cdots$. We then compare the estimated offset for each audio recording to the true offset. Let e denote the difference between the estimated offset (e.g. ΔB_{hyp}) and the true offset (e.g. ΔB_{ref}). If $|e| > \gamma$, where γ specifies an error tolerance, we consider that particular alignment to be incorrect. We can compute the fraction of alignments that are correct at a fixed error tolerance. By sweeping across a range of γ values, we can characterize the tradeoff between accuracy and error tolerance.

Note that an alignment scenario with K audio recordings will generate $K-1$ predictions that are either correct or incorrect. Our accuracy versus error tolerance curves aggregate the results of these predictions over all alignment scenarios. In addition to the accuracy versus error tolerance tradeoff, we can also succinctly characterize the performance of a system by looking at the accuracy at a fixed error tolerance.

It is useful to point out that the anchor file for scoring and the anchor file in our alignment system (as described previously) are totally independent concepts. Our evaluation metric should not depend on our selection of scoring anchor file, since this selection is arbitrary. Accordingly, for each alignment scenario, we consider all possible channels as the scoring anchor file, and choose the one which yields the highest accuracy. This step is necessary to prevent an unlucky selection from unfairly penalizing the results. For example, if the scoring anchor file is aligned incorrectly, the $N-1$ predicted alignments will all be incorrect, even if the other $N-1$ files are aligned correctly amongst themselves. By considering all possible scoring anchor files, this situation would (correctly) yield $N-2$ correct alignments and 1 incorrect alignment.

IV. Results

In this section we present experimental results for our proposed system. As a baseline comparison to our adaptive fingerprint design, we also ran experiments with five other fingerprint designs: the Philips fingerprint [3], the Shazam fingerprint[5] [9], the boosted fingerprints proposed by Ke et al. [28], the MASK fingerprint [18], and the Panako fingerprint [12]. The Philips and Shazam fingerprints are the most well-known approaches in the literature, the work by Ke and colleagues is one of the most highly cited works among those that incorporate boosting into the fingerprint design process, and the MASK and Panako fingerprints are relatively recent works that extend previous approaches to provide greater robustness to changes in pitch and/or tempo. Thus, these five fingerprint designs span a range of different approaches and include both well-known and recent works.

Figure 3 shows the tradeoff between alignment accuracy and error tolerance for the six different fingerprints. To make the comparison as fair as possible, all six fingerprints were evaluated using the same cumulative alignment algorithm. However, there is one important difference to mention. The fine alignment step described in section IIIC assumes that fingerprints are computed at every time frame and that the Hamming distance between fingerprints corresponds to a measure of dissimilarity. For the three approaches that do not satisfy these assumptions – Shazam, MASK, and Panako – the fine alignment step was omitted. For the experiments, we used the default parameter settings in the provided implementation or reference paper. The proposed adaptive fingerprint in Figure 3 uses 16 bits and 32 frames of context.[6] The ordering of the legend corresponds to the ordering of performance at 100ms error tolerance.

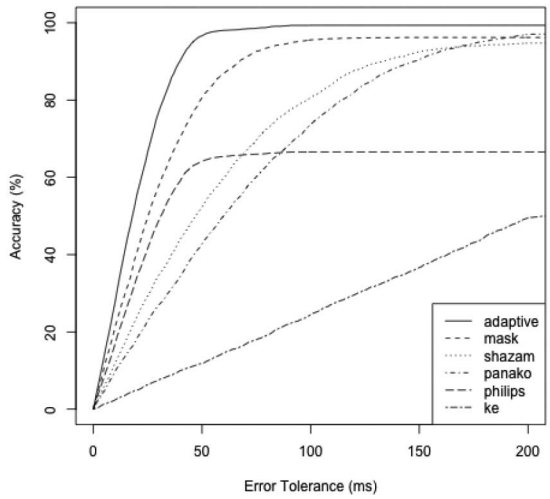

Fig. 3. The tradeoff between accuracy and error tolerance for six different fingerprints. The ordering of the legend corresponds to the performance ordering at 100 ms error threshold.

There are three things to notice about Figure 3. First, there are two separate dimensions along which we can measure the performance of a fingerprint design: precision and robustness. Precision corresponds to how quickly the performance curve levels off, and robustness refers to the accuracy at which the curve levels off. It is important to point out that these two dimensions are not necessarily correlated. Some fingerprints have high precision but low robustness, such as the Philips fingerprint. Other fingerprints have high robustness but low precision, such as the Shazam and Panako fingerprints. Both dimensions are important to consider in evaluating the effectiveness of a fingerprint design.

Second, the *relative* performance of the proposed system is very good. Among the six fingerprint designs that were evaluated, the adaptive fingerprint has the best performance both in terms of precision and robustness. The adaptive fingerprint is approximately tied with the Philips fingerprint for highest precision – they both level off at an error tolerance of around 50 ms. It is also slightly more robust than the three other highly robust fingerprints: Shazam, MASK, and Panako. It is interesting that all three of these other approaches level off at approximately the same accuracy, perhaps because all three approaches focus on identifying the location of spectral peaks. While different fingerprints may offer different tradeoffs between precision and robustness, the decision here is clear: the adaptive fingerprints are best along both dimensions.

Third, the *absolute* performance of the proposed system is very good. Beyond simply performing well relative to other

[5]For the Shazam fingerprint, we used the implementation provided by Dan Ellis [64].

[6]The choice of 16 bits is discussed in Section V-A.

fingerprints, the adaptive fingerprint has very good absolute performance numbers. As seen in Figure 3, the adaptive fingerprint achieves an accuracy of 99.4% for 100 ms error tolerance. The errors from this system generally came from close-talking microphone channels that contained only silence (e.g. the channel was not used or the person was silent) or local noise (e.g. the person wore the headset too closely and the microphone picked up their breathing patterns). Furthermore, we can improve the robustness of the alignment even more (if needed) by providing more context information to the adaptive fingerprint and by reducing the number of fingerprint bits in the lookup. We will explore the effect of these two factors on fingerprint robustness in the analysis section. We simply note here, however, that the proposed system works very reliably and robustly.

V. Analysis

In this section we will investigate and answer six questions of interest. These investigations will develop intuition and understanding of the inner workings, capabilities, and limitations of the proposed adaptive fingerprint.

A. Effect of Number of Lookup Bits

The first question we will answer is, "How does the number of lookup bits affect the robustness of the fingerprint?" In many other works, fingerprints are often characterized by 32 bits so that each fingerprint can be represented compactly as a single 32-bit integer. Here, we investigate how the number of bits affects robustness.

Before presenting any experimental results, we can approach the question from a theoretical standpoint. Note that using a higher number of lookup bits results in higher specificity but lower accuracy. To see this, consider a 32-bit fingerprint whose bits are uncorrelated and balanced (each bit is 1 half the time and 0 half the time). If each bit independently has a $\alpha = 90\%$ probability of being correct given a noisy true match, then the fingerprint would be correct $(.9)^{32} \approx 3.4\%$ of the time. When compared to a randomly selected fingerprint, we would expect a random match approximately $\frac{1}{2^{32}}$ of the time. This corresponds roughly to one spurious match for every $10,000$ hours of audio. Clearly, this is far more specificity than we need for our application of interest, which involves a few tens of hours of audio at most. Now consider a 16-bit fingerprint in the same hypothetical scenario. This fingerprint would be correct $(.9)^{16} \approx 18.5\%$ of the time, and it would have roughly one spurious match for every 10 minutes of audio. This is a much more reasonable choice for our application of interest. The tradeoff essentially comes down to this: reducing the number of lookup bits by 1 increases the fingerprint true match accuracy by a factor of α (which was .9 in the example above) but also increases the number of spurious matches by a factor of 2.

Now that we know what the results should look like, we present our experimental results investigating the effect of the number of lookup bits. Figure 4 shows the accuracy (at a fixed 100 ms error tolerance) of the adaptive fingerprint for three different lookup key sizes: 16, 24, and 32 bits. We did not run experiments with an 8-bit lookup since processing the large number of spurious matches would result in very long run times. Each group of bars compares the effect of lookup key size on an adaptive fingerprint with a fixed amount of context, where we consider 2, 8, and 32 frames of context information.

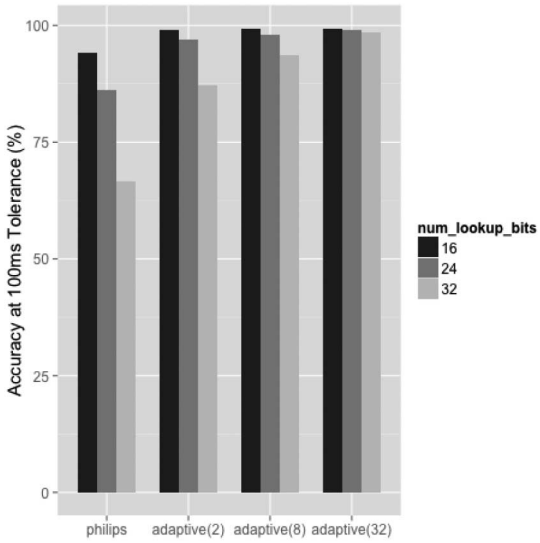

Fig. 4. Determining the effect of the number of fingerprint lookup bits. The leftmost group shows the performance of the Philips fingerprint with a 16-, 24-, and 32-bit lookup. The three rightmost groups show the same comparison for the adaptive fingerprints with 2, 8, and 32 frames of context.

As we expect, reducing the number of fingerprint lookup bits improves system accuracy. This improvement is dramatic for the Philips fingerprint, since the original 32-bit fingerprint leaves a lot of room for improvement. For the more robust adaptive fingerprints, there is still a clear but less dramatic improvement, since the results are nearly saturated already. We can also see the effect of context from this figure, but we will defer discussion of this until a later analysis subsection.

B. Effect of Overlap

The second question we will answer is, "How much temporal overlap is required to correctly align files?" This question will help us understand the conditions necessary to align recordings correctly in meeting scenarios.

To answer this question, we created a slightly different experimental setup in order to isolate the effect of temporal overlap. This modified setup makes two changes to the main experimental setup described previously. First, only two randomly selected channels (rather than all channels) are used to generate each query scenario. This simplifies the setup and allows us to focus on the effect of the amount of overlap between two recordings. Since close-talking microphones often contain extended amounts of silence and we will be considering short amounts of overlap (in the range of a few seconds), we only considered the six tabletop microphone channels for these experiments. Second, we deterministically control the lengths of the two audio segments rather than selecting the lengths randomly as in the previous experiments. Specifically, one channel is selected to be the reference channel and is kept in its entirety

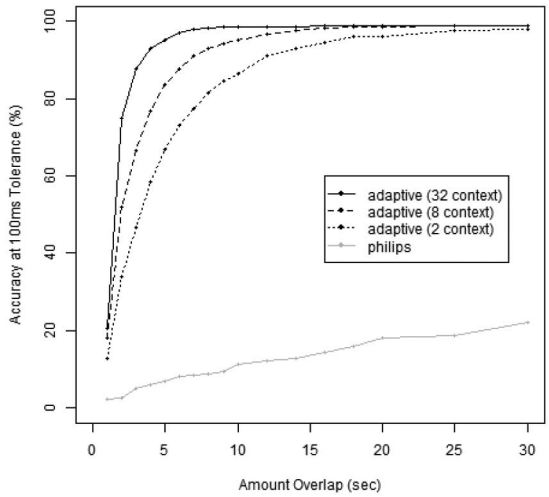

Fig. 5. Determining how much temporal overlap between two recordings is necessary to find a correct alignment. The top three curves show the performance of the adaptive fingerprint with 2, 8, and 32 frames of context. The bottom curve shows the performance of the Philips fingerprint.

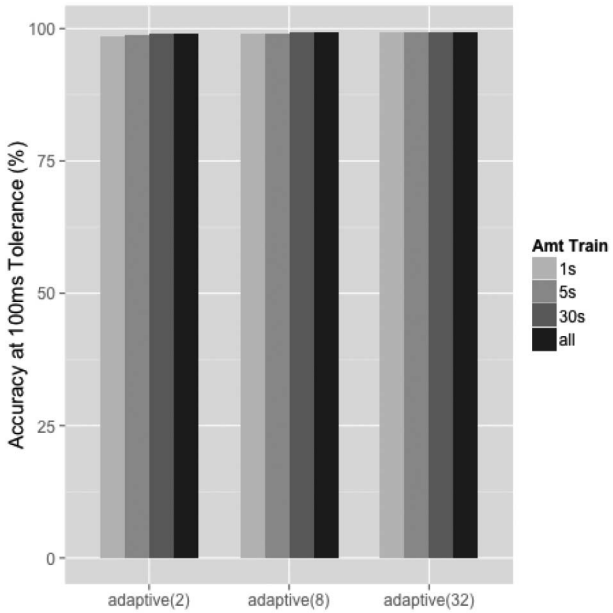

Fig. 6. Determining the effect of the amount of training data. A T-second segment is randomly selected from each channel, and the fingerprint design is learned only on the selected data. Performance is shown for $T = 1$, 5, 30, and ∞ (use all available data). The three groups of bars show the performance of adaptive fingerprints with 2, 8, and 32 frames of context.

(i.e. the entire original audio recording is used). The other channel is selected to be the query, and an N second segment is randomly selected from that channel. Thus, we are given an N second query from one tabletop microphone and we are trying to identify where in the meeting the query occurs in a different tabletop microphone. We can then measure how our accuracy of alignment depends on the length of query. Note that the meetings in the ICSI meeting corpus are typically around an hour long, so with an error tolerance of 100 ms the accuracy of random guessing would be about .006%.

Figure 5 shows the results of our overlap experiments. Each curve shows the effect of query length on the alignment accuracy at a fixed 100 ms error tolerance. Each point on the curve represents the accuracy averaged over approximately 1100 queries. The top three curves correspond to the adaptive fingerprints with 2, 8, and 32 context frames, and the lowest curve corresponds to the Philips fingerprint.

There are two things to notice about the results in Figure 5. First, there is a dramatic improvement in using adaptive fingerprints rather than the Philips fingerprint. For 15 second queries, for example, the adaptive fingerprints have alignment accuracies of 94% and higher, while the Philips fingerprint has an accuracy of 14%. Second, using more context frames improves alignment robustness and shortens the minimum required temporal overlap. Note that the amount of temporal overlap needed to achieve saturated performance decreases as we include more and more context. With 2 context frames, we need about 30 seconds of overlap. With 8 context frames, this number drops to about 15 seconds. With 32 frames of context, 10 seconds of overlap is sufficient to reliably ensure correct alignment. These numbers assume typical meeting dynamics (i.e. the overlap will contain natural pauses but probably not long, extended silence).

C. Effect of Amount of Training Data

The third question we will answer is "How much data is necessary to learn a robust fingerprint design?" When meetings are very long, we would like to know how much data is actually necessary to learn a robust fingerprint design. Alternatively, when meetings are short, we would like to know if the amount of data is sufficient to learn a reasonable fingerprint design.

To answer this question, we ran the original set of experiments with one change: instead of learning the eigenfilters on all of the available data, we selected one T-second random segment from each available channel and learned the eigenfilters only on the selected data. By varying the amount of available training data, we can determine how much data is necessary to learn a robust fingerprint design.

Figure 6 compares the alignment accuracy for adaptive fingerprints with $T =$1, 5, 30, and ∞ (using all available data for training). Each group of bars corresponds to an adaptive fingerprint with a fixed amount of context frames, where we again consider 2, 8, and 32 frames of context.

Surprisingly, even just a 1 second segment from each channel provides enough information to learn a reasonable fingerprint design with good performance. The performance of the adaptive fingerprints is approximately saturated for $T = 5$, so there is only very marginal benefit in using more than 5 seconds of training data from each channel. These results have two very encouraging implications: (1) the adaptive fingerprints will work well for any length of meeting, even very short meetings, and (2) for very long meetings, we can achieve roughly the same level of performance with less computation by only training the filters on a very small subset of data.

D. Effect of Context

The fourth question we will answer is, "How does the amount of context affect fingerprint robustness?" Since we explored a

TABLE I
COMPARING SEVERAL VARIANTS OF THE ALIGNMENT ALGORITHM. THE INDICATED NUMBERS ARE THE ACCURACY AT A SPECIFIED ERROR TOLERANCE. ALL EXPERIMENTS USE 16-BIT ADAPTIVE FINGERPRINTS WITH 2 CONTEXT FRAMES

Alignment Algorithm	Error Tolerance			
	25ms	50ms	75ms	100ms
Pairwise, no refinement	52.1%	82.9%	95.5%	98.7%
Cumulative, no refinement	51.2%	83.3%	95.1%	98.3%
Cumulative, with refinement	66.5%	96.5%	98.5%	99.2%

range of context values in all of our previous analysis experiments, we will simply revisit our previous results but now with a focus on the effect of context.

Earlier we saw that using fewer lookup bits improves accuracy at the expense of computation. This can be seen in Figure 4 by comparing the results within each group of bars. But we can see the effect of context in the same figure by comparing results *across* the groups of bars. For example, if we look at the rightmost bar in each group, we can see that the accuracy increases from 87.2% to 93.5% to 98.4% as we increase the context from 2 to 8 to 32 frames. For fewer lookup bits, we see a similar but less dramatic improvement, since the results are closer to saturation. Clearly, using more context makes the system more robust, though the amount of improvement in system-level accuracy depends on how saturated the results are.

We can similarly revisit the results in Figure 6 with a focus on context. Because so little data is needed to train robust filters, however, we see little differentiation between different amounts of context. For all practical application scenarios, there is more than enough data to learn a robust fingerprint design for up to 32 context frames.

E. Assessing the Alignment Algorithm

The fifth question we will answer is, "How much actual benefit is gained by aggregating cumulative evidence and doing a refined alignment?" We can tease apart the effect of these two components by starting with a pairwise out-of-the-box alignment approach, and then adding in these components one at a time.

Table I compares the performance of three different alignment algorithms. The top line shows the performance of a pairwise alignment approach, similar to the works by Kennedy and Naaman [55] and Su et al. [56]. This approach does not aggregate cumulative evidence and does not do a refined alignment step. The second line shows the performance when we aggregate cumulative evidence, but without a refined alignment. The third line shows the performance when we aggregate cumulative evidence and perform a refined alignment. All experiments use a 16-bit adaptive fingerprint with 2 frames of context.

The results in Table I are somewhat surprising. There is a significant benefit in using our proposed approach over a simple pairwise out-of-the-box approach, but *all* of the benefit is coming from the refined alignment step. Comparing the top two rows of the table, we see that the accuracy is roughly the same. Sometimes the accuracy is slightly higher and sometimes it is slightly lower, depending on the error threshold. But there seems to be no measurable benefit (or detriment) to aggregating cumulative evidence in this scenario. On the other hand, doing a refined alignment yields drastic improvements at all error thresholds. For example, at a 25 ms error threshold, the refined alignment improves the accuracy from about 52% to 66%. As we might expect, the refined alignment improves accuracy the most for very small error thresholds.

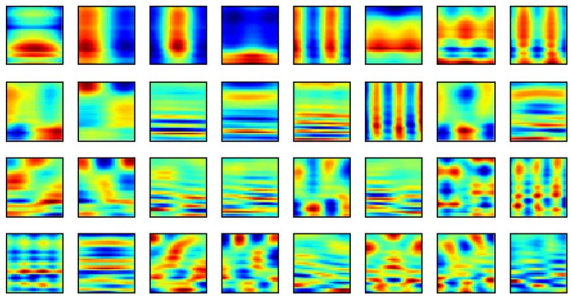

Fig. 7. The top 32 learned eigenfilters from one alignment scenario. The filters are arranged from left to right, and then from top to bottom.

F. Learned Filters

The sixth question we will answer is, "What do the learned filters look like?" The purpose of this question is not so much to improve system performance as it is to gain more intuition and understanding into what constitutes a robust fingerprint design.

Figure 7 shows the top 32 eigenfilters for one particular query scenario when the fingerprint is given 32 context frames. Recall that the filters are learned from scratch on each query, so each query will have its own set of filters. Figure 7 shows one example set of filters. The filters progress from left to right, and then from top to bottom. So the upper leftmost is the first filter, and the lower rightmost is the 32^{nd} filter.

There are three observations to make about the filters shown in Figure 7. First, modulations in both time and frequency are useful. Some of the filters primarily capture modulations in time, such as filters 2, 3, 5, and 8 in Figure 7. Some of the filters primarily capture modulations in frequency, such as filters 1, 4, 6, and 13. Other filters capture modulations in both time and frequency, such as filters 7 and 15. The key thing to point out is that both are important, so we should not emphasize one to the exclusion of the other. For example, giving the adaptive filters only two frames of context forces the filters to focus almost entirely on frequency modulations rather than temporal modulations, since two frames is insufficient to capture much variation in the temporal dimension. Second, low modulations seem to be most important and useful. We can see a progression from low modulations to higher frequency modulations as we get to later and later filters. For example, the second, third, fifth, eighth, and fourteenth filters capture higher and higher frequency modulations in time. In general, we see a progression from slower modulations in the top few filters to faster modulations in later filters. Third, the filters are remarkably consistent from query to query. When we look at the learned filters for many different queries, we observe that the first 8-10 filters are usually very similar and in approximately the same order. As

TABLE II
AVERAGE RUN TIME REQUIRED TO ALIGN K RECORDINGS EACH OF LENGTH L USING THE PROPOSED APPROACH. EXPERIMENTS WERE RUN ON A 2.2 GHZ INTEL XEON PROCESSOR

L\K	2	4	6	8
5 min	17sec	25sec	38sec	52sec
10 min	24sec	42sec	67sec	97sec
20 min	42sec	80sec	139sec	211sec
30 min	60sec	120sec	214sec	340sec
60 min	125sec	272sec	506sec	795sec

TABLE III
THEORETICAL LOWER BOUND ON RUN TIME REQUIRED TO ALIGN K RECORDINGS EACH OF LENGTH L USING A NAIVE PAIRWISE CROSS-CORRELATION APPROACH. ASSUMES 8 KHZ DATA AND ONE ADDITION OR MULTIPLICATION PER CYCLE ON A 2.2 GHZ PROCESSOR.

L\K	2	4	6	8
5 min	1.5hr	8.7hr	22hr	41hr
10 min	5.8hr	35hr	87hr	163hr
20 min	23hr	140hr	349hr	652hr
30 min	52hr	314hr	785hr	1466hr
60 min	209hr	1257hr	3142hr	5865hr

we progress to later filters, there is more diversity and difference from query to query. This suggests that these top few filters are capturing information that is characteristic of the genre of audio data (i.e. meeting speech), rather than something specific to what is being said in a particular meeting.

VI. DISCUSSION

In this section we discuss three practical takeaway lessons from our experimental findings.

Takeaway lesson #1: Adaptive fingerprints are a *robust* way to align meeting recordings. Our system is able to achieve more than 99% alignment accuracy at a reasonable error tolerance (100ms) on alignment scenarios that are much more challenging and aggressive than would typically be found in practice (i.e. shorter audio recordings and less temporal overlap). The files that our system could not align correctly were close-talking microphone channels consisting almost entirely of silence or local noise such as the speaker breathing into the microphone. Only about 10 seconds of typical meeting speech is necessary to identify a correct alignment. Furthermore, since only a few seconds of data from each recording is needed to reliably learn a robust fingerprint design, the adaptive fingerprints can be learned quickly and efficiently on meetings of any length, even short ones. In general, the adaptive fingerprints should be able to align meeting recordings very reliably.

Takeaway lesson #2: Adaptive fingerprints are an *efficient* way to align meeting recordings. We can compare the amount of time required to align files using a simple cross-correlation approach and using our proposed approach. Table II shows the average run time required to align K recordings that are each exactly L minutes long using the proposed approach. So, for example, aligning 8 files that are each 1 hour long takes 795 seconds on average. Here, we have chosen values of K and L that span a range of realistic scenarios. These measurements were taken on a single thread of a 2.2 GHz Intel Xeon processor.

Making similar measurements on a naive pairwise cross-correlation approach would be computationally infeasible. However, we can analyze such an approach to determine a lower bound on actual running time. Computing the cross-correlation at every possible offset between two files requires approximately N^2 multiplications and N^2 additions, where N is the number of time-domain samples. For K recordings, the total number of multiplications (and additions) required would thus be $\binom{K}{2} N^2$. Table III shows a theoretical lower bound on the run time required to align K recordings of length L using a naive pairwise cross-correlation approach, assuming 8kHz data and a simple model of performing a single addition or multiplication per cycle on a 2.2 GHz processor.

There are two things to notice in Tables II and III. First, the pairwise cross-correlation approach is computationally infeasible. Just aligning two 5-minute recordings would require 90 minutes to run. Aligning eight recordings that are each 60 minutes long would require more than 8 months. Second, the running time for the proposed approach is acceptable for any realistic scenario. The running times are on the order of minutes, which is acceptable for an offline task. It would also be trivial to parallelize this task for a further reduction in computation time. When aligning eight recordings that are each 60 minutes long, the proposed approach results in a savings of more than four orders of magnitude.

Takeaway lesson #3: If more robustness is needed, it can be achieved very simply. There are two parameters that we can modify to improve the robustness of the fingerprint, both of which come at the cost of more computation. The first parameter is the amount of context. We saw in section V-D that increasing the amount of context improves the alignment accuracy and reduces the amount of temporal overlap needed for correct alignment. Varying this first parameter increases computation linearly. Note that increasing the amount of context by a factor of N means doing N times as many dense multiplications at each frame when extracting fingerprints. The second parameter is the number of fingerprint lookup bits. We saw earlier that decreasing the number of lookup bits increases the alignment accuracy. Varying this second parameter increases computation exponentially: for every 1 bit reduction in the lookup key size, we will have to process twice as many spurious fingerprint matches.

Given these two parameters, we can adopt the following simple strategy to increase the robustness if needed: First, we increase the context to gain robustness at the expense of a linear increase in computation. If the fingerprint is still not robust enough, we can then begin decreasing the number of fingerprint lookup bits and paying the heavier exponential cost. The number of lookup bits adjusts robustness at a coarse granularity, and the context adjusts robustness at a fine granularity. Because the proposed fingerprints are learned on the fly in an unsupervised manner, there is very little system tuning to do given a new set of data to align. We can start with a reasonable setting (e.g. 16-bit lookup with 32 frames of context) and, if more robustness is needed, we can follow the simple strategy above.

VII. Conclusion

We have proposed a method for aligning a set of overlapping meeting recordings. Our method uses an audio fingerprint representation based on spectrotemporal eigenfilters that are learned on-the-fly in an unsupervised manner. The recordings are then aligned with an iterative, greedy two-stage alignment algorithm which performs a rough alignment using indexing techniques, followed by a fine-grained alignment based on Hamming distance. Using the ICSI meeting corpus as source material to generate challenging alignment scenarios, our proposed method is able to achieve greater than 99% alignment accuracy at a reasonable error tolerance of 0.1 seconds. The method only requires a few seconds of audio from each channel to learn a robust fingerprint design, and can robustly identify an alignment with 10 seconds of temporal overlap in a typical meeting scenario. One of the greatest benefits of our approach is that it requires little to no system tuning given a new set of recordings to align, since the fingerprint is learned on-the-fly. We demonstrated how the robustness of the fingerprint can be improved by increasing the amount of context information or by decreasing the number of fingerprint lookup bits. We have discussed the tradeoffs of changing these two factors and proposed a simple strategy to adjust them in practice. Future work includes testing this approach on different types of data besides meeting recordings, including other speech genres and music.

Acknowledgment

Thanks to Adam Janin, Nelson Morgan, Steven Wegmann, and Eric Chu for constructive feedback and discussions.

References

[1] Shazam. (2014). [Online]. Available: http://www.shazam.com/
[2] SoundHound. (2014). [Online]. Available: http://www.soundhound.com/
[3] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system," in *Proc. Int. Soc. Music Inf. Retrieval (ISMIR'02)*, Paris, France, Oct. 2002, pp. 107–115.
[4] B. Coover and J. Han, "A power mask based audio fingerprint," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP'14)*, 2014, pp. 1394–1398.
[5] J. S. Seo, "An asymmetric matching method for a robust binary audio fingerprinting," *IEEE Signal Process. Lett.*, vol. 21, no. 7, pp. 844–847, Jul. 2014.
[6] H. Schreiber, P. Grosche, and M. Müller, "A re-ordering strategy for accelerating index-based audio fingerprinting," in *Proc. Int. Soc. Music Inf. Retrieval (ISMIR'11)*, 2011, pp. 127–132.
[7] G. Yang, X. Chen, and D. Yang, "Efficient music identification by utilizing space-saving audio fingerprinting system," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME'14)*, 2014, pp. 1–6.
[8] C. Yu, R. Wang, J. Xiao, and J. Sun, "High performance indexing for massive audio fingerprint data," *IEEE Trans. Consum. Electron.*, vol. 60, no. 4, pp. 690–695, Nov. 2014.
[9] A. L.-C. Wang, "An industrial-strength audio search algorithm," in *Proc. Int. Soc. Music Inf. Retrieval (ISMIR'03)*, Baltimore, MD, USA, Oct. 2003, pp. 7–13.
[10] J. George and A. Jhunjhunwala, "Scalable and robust audio fingerprinting method tolerable to time-stretching," in *Proc. IEEE Int. Conf. Digit. Signal Process. (DSP)*, 2015, pp. 436–440.
[11] S. Fenet, G. Richard, and Y. Grenier, "A scalable audio fingerprint method with robustness to pitch-shifting," in *Proc. Int. Soc. Music Inf. Retrieval (ISMIR'11)*, 2011, pp. 121–126.
[12] J. Six and M. Leman, "Panako: A scalable acoustic fingerprinting system handling time-scale and pitch modification," in *Proc. Int. Soc. Music Inf. Retrieval (ISMIR'14)*, 2014.
[13] R. Sonnleitner and G. Widmer, "Quad-based audio fingerprinting robust to time and frequency scaling," in *Proc. Int. Conf. Digit. Audio Effects*, 2014, pp. 173–180.
[14] S. Baluja and M. Covell, "Audio fingerprinting: Combining computer vision & data stream processing," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP'07)*, Honolulu, HI, USA, Apr. 2007, pp. 213–216.
[15] S. Baluja and M. Covell, "Waveprint: Efficient wavelet-based audio fingerprinting," *Pattern Recognit.*, vol. 41, no. 11, pp. 3467–3480, May 2008.
[16] Y. Shi, W. Zhang, and J. Liu, "Robust audio fingerprinting based on local spectral luminance maxima scheme," in *Proc. Interspeech*, 2011, pp. 2485–2488.
[17] C. V. Cotton and D. P. Ellis, "Audio fingerprinting to identify multiple videos of an event," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP'10)*, 2010, pp. 2386–2389.
[18] X. Anguera, A. Garzon, and T. Adamek, "MASK: Robust local features for audio fingerprinting," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME'12)*, Melbourne, VIC, Australia, Jul. 2012, pp. 455–460.
[19] M. Ramona and G. Peeters, "Audioprint: An efficient audio fingerprint system based on a novel cost-less synchronization scheme," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP'13)*, 2013, pp. 818–822.
[20] S. Fenet, Y. Grenier, and G. Richard, "An extended audio fingerprint method with capabilities for similar music detection," in *Proc. Int. Soc. Music Inf. Retrieval (ISMIR'13)*, 2013, pp. 569–574.
[21] M. Malekesmaeili and R. K. Ward, "A local fingerprinting approach for audio copy detection," *Signal Process.*, vol. 98, pp. 308–321, 2014.
[22] S. Sukittanon and L. E. Atlas, "Modulation frequency features for audio fingerprinting," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP'02)*, 2002, pp. 1773–1776.
[23] M. Ramona and G. Peeters, "Audio identification based on spectral modeling of bark-bands energy and synchronization through onset detection," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP'11)*, 2011, pp. 477–480.
[24] J. Herre, E. Allamanche, and O. Hellmuth, "Robust matching of audio signals using spectral flatness features," in *Proc. IEEE Workshop Appl. Signal Process. Audio Acoust.*, New Platz, NY, USA, Oct. 2001, pp. 127–130.
[25] E. Allamanche, J. Herre, O. Hellmuth, B. Fröba, T. Kastner, and M. Cremer, "Content-based identification of audio material using MPEG-7 low level description," in *Proc. Int. Soc. Music Inf. Retrieval (ISMIR'01)*, 2001.
[26] J. S. Seo, M. Jin, S. Lee, D. Jang, S. Lee, and C. D. Yoo, "Audio fingerprinting based on normalized spectral subband centroids," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP'05)*, 2005, pp. 213–216.
[27] J. S. Seo, M. Jin, S. Lee, D. Jang, S. Lee, and C. D. Yoo, "Audio fingerprinting based on normalized spectral subband moments," *IEEE Signal Process. Lett.*, vol. 13, no. 4, pp. 209–212, Apr. 2006.
[28] Y. Ke, D. Hoiem, and R. Sukthankar, "Computer vision for music identification," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog. (CVPR'05)*, San Diego, CA, USA, Jun. 2005, pp. 597–604.
[29] P. Viola and M. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, Jul. 2004.
[30] D. Jang, C. D. Yoo, S. Lee, S. Kim, and T. Kalker, "Pairwise boosted audio fingerprint," *IEEE Trans. Inf. Forensics Security*, vol. 4, no. 4, pp. 995–1004, 2009.
[31] S. Kim and C. D. Yoo, "Boosted binary audio fingerprint based on spectral subband moments," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP'07)*, Honolulu, HI, USA, Apr. 2007, pp. 241–244.
[32] C. J. C. Burges, J. C. Platt, and S. Jana, "Distortion discriminant analysis for audio fingerprinting," *IEEE Trans. Speech Audio Process.*, vol. 11, no. 3, pp. 165–174, May 2003.
[33] P. Over *et al.*, "TRECVID 2011—An overview of the goals, tasks, data, evaluation mechanisms and metrics," in *Proc. TREC Video Retrieval Eval. Online (TRECVID)*, Gaithersburg, MD, USA, Dec. 2011.
[34] C. W. Ngo *et al.*, "VIREO/DVMM at TRECVID 2009: High-level feature extraction, automatic video search, and content-based copy detection," in *Proc. TRECVID*, 2009, pp. 415–432.
[35] M. Héritier, V. Gupta, L. Gagnon, G. Boulianne, and S. Foucher, and P. Cardinal, "CRIM's content-based copy detection system for TRECVID," in *Proc. TRECVID Workshop*, Gaithersburg, MD, USA, 2009.
[36] A. Saracoglu *et al.*, "Content based copy detection with coarse audio-visual fingerprints," in *Proc. IEEE Int. Workshop Content-Based Multimedia Indexing (CBMI'09)*, 2009, pp. 213–218.

[37] V. N. Gupta, G. Boulianne, and P. Cardinal, "CRIM's content-based audio copy detection system for TRECVID 2009," *Multimedia Tools Appl.*, vol. 60, no. 2, pp. 371–387, 2012.
[38] E. Younessian, X. Anguera, T. Adamek, N. Oliver, and D. Marimon, "Telefonica Research at TRECVID 2010 content-based copy detection," in *Proc. TRECVID*, 2010.
[39] Y. Uchida, S. Sakazawa, M. Agrawal, and M. Akbacak, "KDDI Labs and SRI International at TRECVID 2010: Content-based copy detection," in *Proc. TRECVID*, 2010.
[40] R. Mukai, T. Kurozumi, K. Hiramatsu, T. Kawanishi, H. Nagano, and K. Kashino, "NTT Communication Science Laboratories at TRECVID 2010content based copy detection," in *Proc. TRECVID*, 2010.
[41] Y. Liu, W.-L. Zhao, C.-W. Ngo, C.-S. Xu, and H.-Q. Lu, "Coherent bag-of audio words model for efficient large-scale video copy detection," in *Proc. ACM Int. Conf. Image Video Retrieval*, 2010, pp. 89–96.
[42] C. Ouali, P. Dumouchel, and V. Gupta, "A robust audio fingerprinting method for content-based copy detection," in *Proc. IEEE Int. Workshop Content-Based Multimedia Indexing (CBMI)*, 2014, pp. 1–6.
[43] H. Jégou, J. Delhumeau, J. Yuan, G. Gravier, and P. Gros, "BABAZ: A large scale audio search system for video copy detection," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP'12)*, 2012, pp. 2369–2372.
[44] H. Khemiri, D. Petrovska-Delacretaz, and G. Chollet, "Detection of repeating items in audio streams using data-driven ALISP sequencing," in *Proc. IEEE Int. Conf. Adv. Technol. Signal Image Process. (ATSIP)*, 2014, pp. 446–451.
[45] S. Fenet, M. Moussallam, Y. Grenier, G. Richard, and L. Daudet, "A framework for fingerprint-based detection of repeating objects in multimedia streams," in *Proc. IEEE 20th Eur. Signal Process. Conf. (EUSIPCO)*, 2012, pp. 1464–1468.
[46] R. Radhakrishnan and W. Jiang, "Repeating segment detection in songs using audio fingerprint matching," in *Proc. IEEE Asia-Pac. Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA ASC)*, 2012, pp. 1–5.
[47] C. Herley, "ARGOS: Automatically extracting repeating objects from multimedia streams," *IEEE Trans. Multimedia*, vol. 8, no. 1, pp. 115–129, Feb. 2006.
[48] J. P. Ogle and D. P. W. Ellis, "Fingerprinting to identify repeated sound events in long-duration personal audio recordings," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP'07)*, Honolulu, HI, USA, Apr. 2007, pp. 233–236.
[49] I. Bisio, A. Delfino, F. Lavagetto, and M. Marchese, "A television channel real-time detector using smartphones," *IEEE Trans. Mobile Comput.*, vol. 14, no. 1, pp. 14–27, Jan. 2015.
[50] N. Q. Duong and F. Thudor, "Movie synchronization by audio landmark matching," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP'13)*, 2013, pp. 3632–3636.
[51] N. Q. Duong, C. Howson, and Y. Legallais, "Fast second screen TV synchronization combining audio fingerprint technique and generalized cross correlation," in *Proc. IEEE Int. Conf. Consum. Electron. Berlin (ICCE)*, 2012, pp. 241–244.
[52] R. Macrae, X. Anguera, and N. Oliver, "MuViSync: Realtime music video alignment," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, 2010, pp. 534–539.
[53] T.-K. Hon, L. Wang, J. D. Reiss, and A. Cavallaro, "Audio fingerprinting for multi-device self-localization," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 23, no. 10, pp. 1623–1636, Oct. 2015.
[54] P. Shrestha, M. Barbieri, and H. Weda, "Synchronization of multi-camera video recordings based on audio," in *Proc. ACM Int. Conf. Multimedia*, 2007, pp. 545–548.
[55] L. Kennedy and M. Naaman, "Less talk, more rock: Automated organization of community-contributed collections of concert videos," in *Proc. ACM Int. Conf. World Wide Web*, 2009, pp. 311–320.
[56] K. Su, M. Naaman, A. Gurjar, M. Patel, and D. P. W. Ellis, "Making a scene: Alignment of complete sets of clips based on pairwise audio match," in *Proc. ACM Int. Conf. Multimedia Retrieval (ICMR'12)*, 2012.
[57] N. J. Bryan, P. Smaragdis, and G. J. Mysore, "Clustering and synchronizing multi-camera video via landmark cross-correlation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP'12)*, 2012, pp. 2389–2392.
[58] J. Six and M. Leman, "Synchronizing multimodal recordings using audio-to-audio alignment," *J. Multimodal User Interfaces*, vol. 9, no. 3, pp. 223–229, 2015.
[59] T. Tsai and A. Stolcke, "Aligning meeting recordings via adaptive fingerprinting," in *Proc. Interspeech*, 2015, pp. 786–790.
[60] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. 20th Annu. Symp. Comput. Geom.*, 2004, pp. 253–262.
[61] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. 21st Adv. Neural Inf. Process. Syst. (NIPS'09)*, 2009, pp. 1753–1760.
[62] A. Janin, *et al.*, "The ICSI meeting corpus," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.* (ICASSP'03), 2003, pp. 364–367.
[63] J. Carletta, "Unleashing the killer corpus: Experiences in creating the multi-everything AMI meeting corpus," *Lang. Resour. Eval.*, vol. 41, no. 2, pp. 181–190, 2007.
[64] D. Ellis. (2015). *Robust Landmark-Based Audio Fingerprinting* [Online]. Available: http://labrosa.ee.columbia.edu/matlab/fingerprint/

T. J. Tsai (S'13) received the B.S. and M.S. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 2006 and 2007, respectively. He is currently pursuing the Ph.D. degree in the electrical engineering and computer science at the University of California Berkeley, Berkeley, CA, USA (in joint collaboration with the International Computer Science Institute). From 2008 to 2010, he worked at SoundHound, a startup that allows users to search for music by singing, humming, or playing a recorded track.

Andreas Stolcke (M'95–SM'05–F'11) received the Ph.D. degree in computer science from the University of California, Berkeley., Berkeley, CA, USA. He subsequently worked as a Senior Research Engineer with the Speech Technology and Research Laboratory, SRI International, Menlo Park, CA, USA, and is currently a Principal Researcher with the Speech and Dialog Research Group at Microsoft Research in Mountain View, CA; he is also an External Fellow at the International Computer Science Institute in Berkeley, CA, USA. He is also the author of a widely used open-source toolkit for statistical language modeling. He research interests include machine language learning, parsing, speech recognition, speaker recognition, and speech understanding.