An Industrial-Strength Audio Search Algorithm

Avery Li-Chun Wang
avery@shazamteam.com
Shazam Entertainment, Ltd.

USA:
2925 Ross Road
Palo Alto, CA 94303

United Kingdom:
375 Kensington High Street
4th Floor Block F
London W14 8Q

We have developed and commercially deployed a flexible audio search engine. The algorithm is noise and distortion resistant, computationally efficient, and massively scalable, capable of quickly identifying a short segment of music captured through a cellphone microphone in the presence of foreground voices and other dominant noise, and through voice codec compression, out of a database of over a million tracks. The algorithm uses a combinatorially hashed time-frequency constellation analysis of the audio, yielding unusual properties such as transparency, in which multiple tracks mixed together may each be identified. Furthermore, for applications such as radio monitoring, search times on the order of a few milliseconds per query are attained, even on a massive music database.

1 Introduction

Shazam Entertainment, Ltd. was started in 2000 with the idea of providing a service that could connect people to music by recognizing music in the environment by using their mobile phones to recognize the music directly. The algorithm had to be able to recognize a short audio sample of music that had been broadcast, mixed with heavy ambient noise, subject to reverb and other processing, captured by a little cellphone microphone, subjected to voice codec compression, and network dropouts, all before arriving at our servers. The algorithm also had to perform the recognition quickly over a large database of music with nearly 2M tracks, and furthermore have a low number of false positives while having a high recognition rate.

This was a hard problem, and at the time there were no algorithms known to us that could satisfy all these constraints. We eventually developed our own technique that met all the operational constraints [1].

We have deployed the algorithm to scale in our commercial music recognition service, with over 1.8M tracks in the database. The service is currently live in Germany, Finland, and the UK, with over a half million users, and will soon be available in additional countries in Europe, Asia, and the USA. The user experience is as follows: A user hears music playing in the environment. She calls up our service using her mobile phone and samples up to 15 seconds of audio. An identification is performed on the sample at our server, then the track title and artist are sent back to the user via SMS text messaging. The information is also made available on a web site, where the user may register and log in with her mobile phone number and password. At the web site, or on a smart phone, the user may view her tagged track list and buy the CD. The user may also download the ringtone corresponding to the tagged track, if it is available. The user may also send a 30-second clip of the song to a friend. Other services, such as purchasing an MP3 download may become available soon.

A variety of similar consumer services has sprung up recently. Musiwave has deployed a similar mobile-phone music identification service on the Spanish mobile carrier Amena using Philips' robust hashing algorithm [2-4]. Using the algorithm from Relatable, Neuros has included a sampling feature on their MP3 player which allows a user to collect a 30-second sample from the built-in radio, then later plug into an online server to identify the music [5,6]. Audible Magic uses the Muscle Fish algorithm to offer the Clango service for identifying audio streaming from an internet radio station [7-9].

The Shazam algorithm can be used in many applications besides just music recognition over a mobile phone. Due to the ability to dig deep into noise we can identify music hidden behind a loud voiceover, such as in a radio advert. On the other hand, the algorithm is also very fast and can be used for copyright monitoring at a search speed of over 1000 times realtime, thus enabling a modest server to monitor significantly many media streams. The algorithm is also suitable for content-based cueing and indexing for library and archival uses.

2 Basic principle of operation

Each audio file is "fingerprinted," a process in which reproducible hash tokens are extracted. Both "database" and "sample" audio files are subjected to the same analysis. The fingerprints from the unknown sample are matched against a large set of fingerprints derived from the music database. The candidate matches are subsequently evaluated for correctness of match. Some guiding principles for the attributes to use as fingerprints are that they should be temporally localized, translation-invariant, robust, and sufficiently entropic. The temporal locality

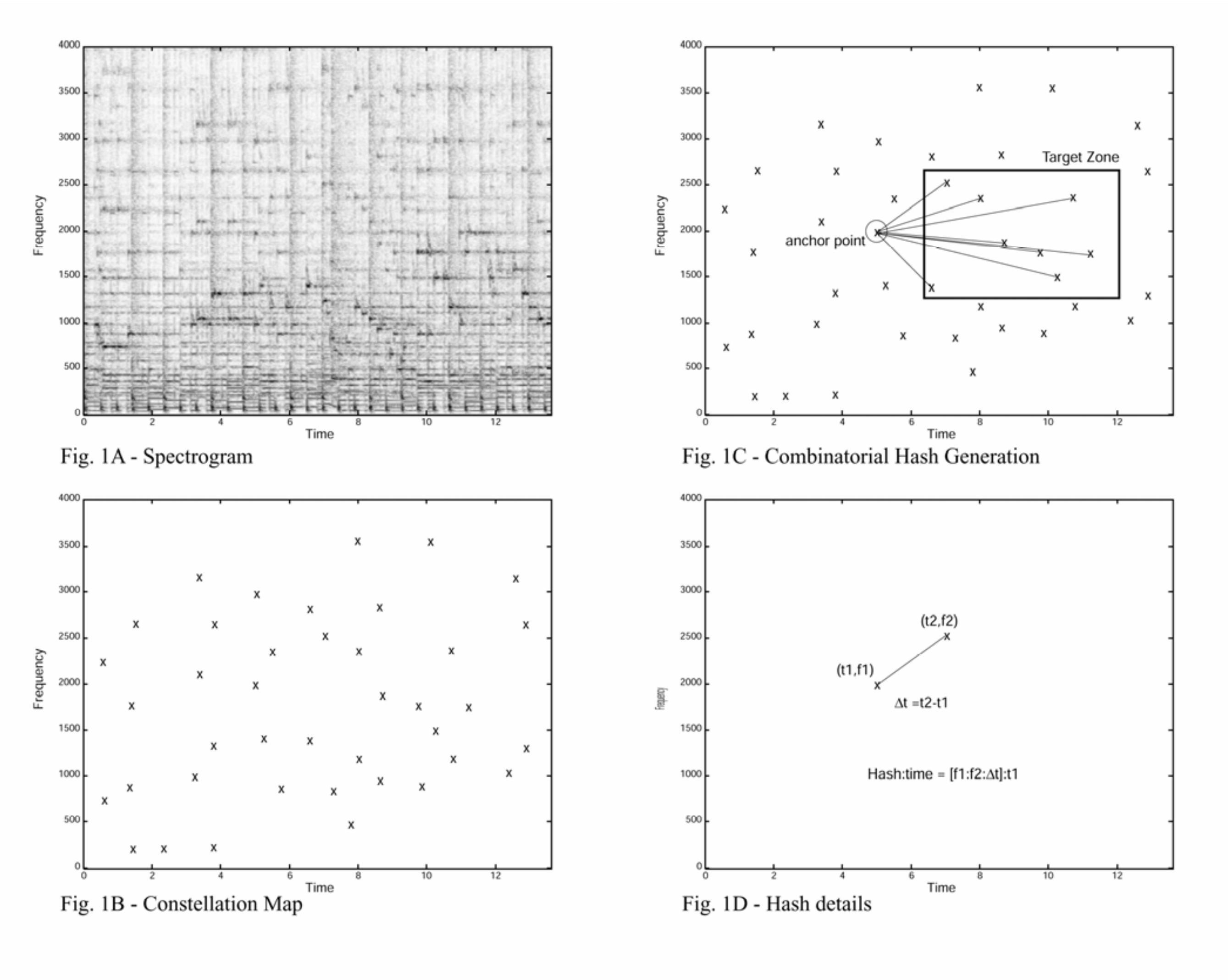


Fig. 1A - Spectrogram

Fig. 1C - Combinatorial Hash Generation

Fig. 1B - Constellation Map

Fig. 1D - Hash details

guideline suggests that each fingerprint hash is calculated using audio samples near a corresponding point in time, so that distant events do not affect the hash. The translation-invariant aspect means that fingerprint hashes derived from corresponding matching content are reproducible independent of position within an audio file, as long as the temporal locality containing the data from which the hash is computed is contained within the file. This makes sense, as an unknown sample could come from any portion of the original audio track. Robustness means that hashes generated from the original clean database track should be reproducible from a degraded copy of the audio. Furthermore, the fingerprint tokens should have sufficiently high entropy in order to minimize the probability of false token matches at non-corresponding locations between the unknown sample and tracks within the database. Insufficient entropy leads to excessive and spurious matches at non-corresponding locations, requiring more processing power to cull the results, and too much entropy usually leads to fragility and non-reproducibility of fingerprint tokens in the presence of noise and distortion.

There are 3 main components, presented in the next sections.

2.1 Robust Constellations

In order to address the problem of robust identification in the presence of highly significant noise and distortion, we experimented with a variety of candidate features that could survive GSM encoding in the presence of noise. We settled on spectrogram peaks, due to their robustness in the presence of noise and approximate linear superposability [1]. A time-frequency point is a candidate peak if it has a higher energy content than all its neighbors in a region centered around the point. Candidate peaks are chosen according to a density criterion in order to assure that the time-frequency strip for the audio file has reasonably uniform coverage. The peaks in each time-frequency locality are also chosen according amplitude, with the justification that the highest amplitude peaks are most likely to survive the distortions listed above.

Thus, a complicated spectrogram, as illustrated in Figure 1A may be reduced to a sparse set of coordinates, as illustrated in Figure 1B. Notice that at this point the amplitude component has been eliminated. This reduction has the advantage of being fairly insensitive to EQ, as

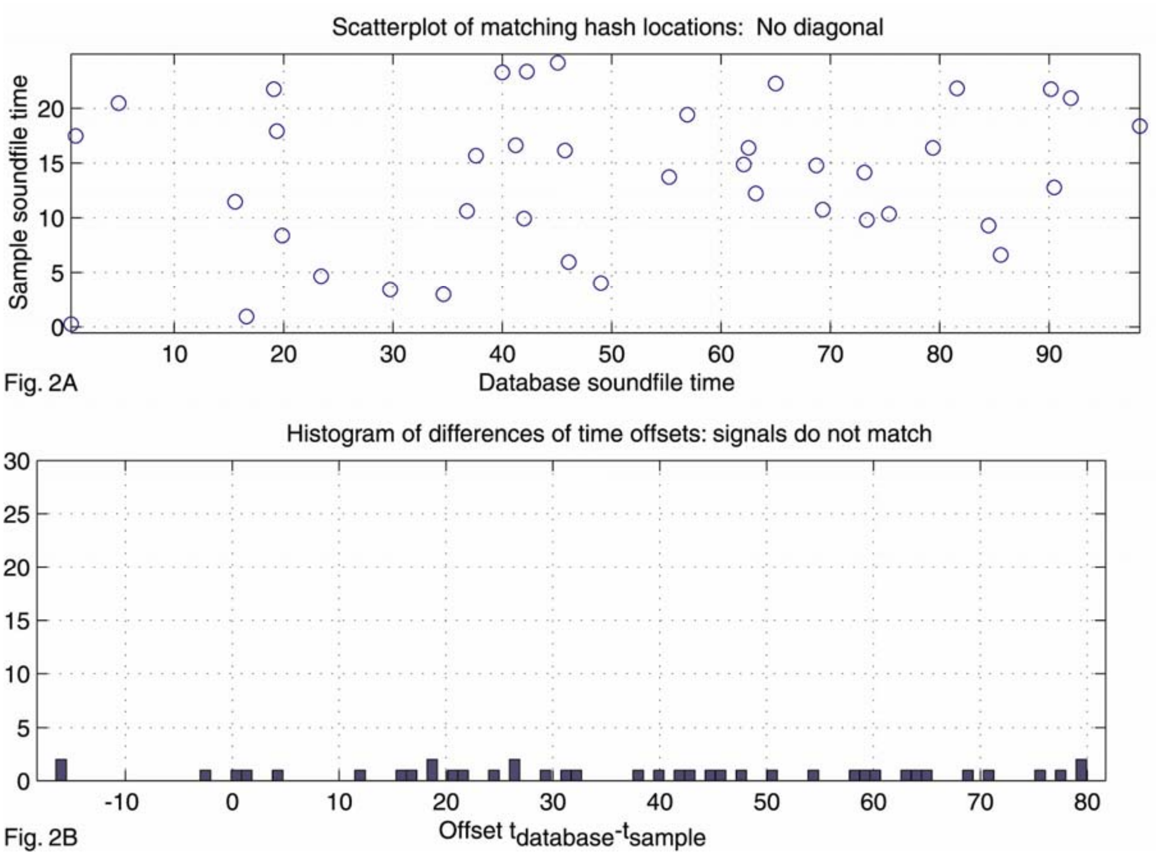

Scatterplot of matching hash locations: No diagonal

Fig. 2A

Histogram of differences of time offsets: signals do not match

Fig. 2B

generally a peak in the spectrum is still a peak with the same coordinates in a filtered spectrum (assuming that the derivative of the filter transfer function is reasonably small—peaks in the vicinity of a sharp transition in the transfer function are slightly frequency-shifted). We term the sparse coordinate lists "constellation maps" since the coordinate scatter plots often resemble a star field.

The pattern of dots should be the same for matching segments of audio. If you put the constellation map of a database song on a strip chart, and the constellation map of a short matching audio sample of a few seconds length on a transparent piece of plastic, then slide the latter over the former, at some point a significant number of points will coincide when the proper time offset is located and the two constellation maps are aligned in register.

The number of matching points will be significant in the presence of spurious peaks injected due to noise, as peak positions are relatively independent; further, the number of matches can also be significant even if many of the correct points have been deleted. Registration of constellation maps is thus a powerful way of matching in the presence of noise and/or deletion of features. This procedure reduces the search problem to a kind of "astronavigation," in which a small patch of time-frequency constellation points must be quickly located within a large universe of points in a strip-chart universe with dimensions of bandlimited frequency versus nearly a billion seconds in the database.

Yang also considered the use of spectrogram peaks, but employed them in a different way [10].

2.2 *Fast Combinatorial Hashing*

Finding the correct registration offset directly from constellation maps can be rather slow, due to raw constellation points having low entropy. For example, a 1024-bin frequency axis yields only at most 10 bits of frequency data per peak. We have developed a fast way of indexing constellation maps.

Fingerprint hashes are formed from the constellation map, in which pairs of time-frequency points are combinatorially associated. Anchor points are chosen, each anchor point having a target zone associated with it. Each anchor point is sequentially paired with points within its target zone, each pair yielding two frequency components plus the time difference between the points (Figure 1C and 1D). These hashes are quite reproducible, even in the presence of noise and voice codec compression. Furthermore, each hash can be packed into a 32-bit unsigned integer. Each hash is also associated with the time offset from the beginning of the respective file to its anchor point, though the absolute time is not a part of the hash itself.

To create a database index, the above operation is carried out on each track in a database to generate a corresponding list of hashes and their associated offset times. Track IDs may also be appended to the small data structs, yielding an

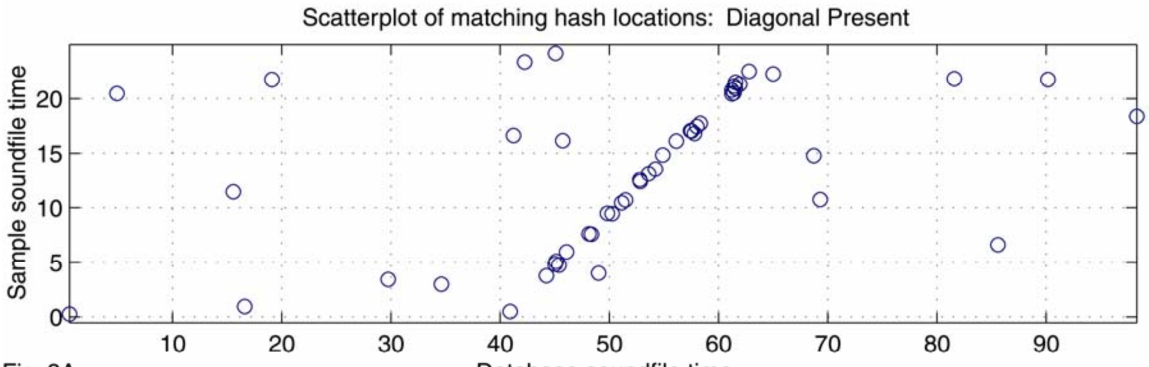


Fig. 3A

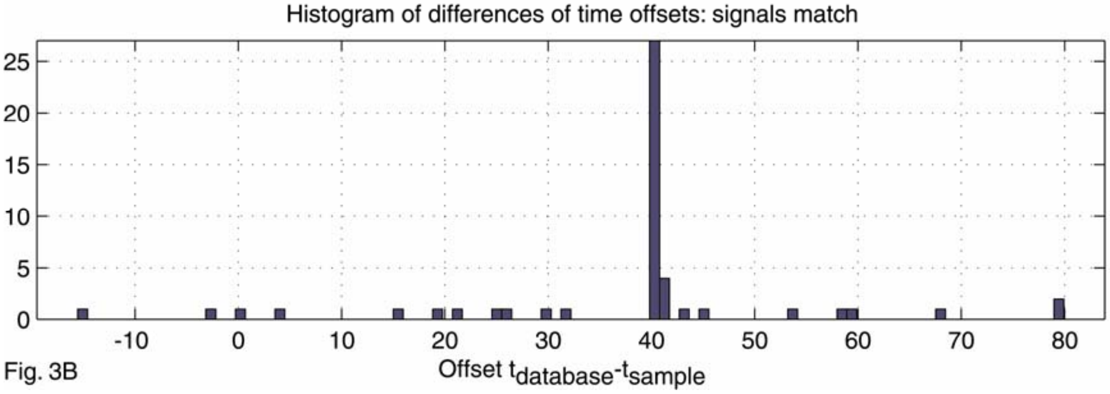


Fig. 3B

aggregate 64-bit struct, 32 bits for the hash and 32 bits for the time offset and track ID. To facilitate fast processing, the 64-bit structs are sorted according to hash token value.

The number of hashes per second of audio recording being processed is approximately equal to the density of constellation points per second times the fan-out factor into the target zone. For example, if each constellation point is taken to be an anchor point, and if the target zone has a fan-out of size F=10, then the number of hashes is approximately equal to F=10 times the number of constellation points extracted from the file. By limiting the number of points chosen in each target zone, we seek to limit the combinatorial explosion of pairs. The fan-out factor leads directly to a cost factor in terms of storage space.

By forming pairs instead of searching for matches against individual constellation points we gain a tremendous acceleration in the search process. For example, if each frequency component is 10 bits, and the Δt component is also 10 bits, then matching a pair of points yields 30 bits of information, versus only 10 for a single point. Then the specificity of the hash would be about a million times greater, due to the 20 extra bits, and thus the search speed for a single hash token is similarly accelerated. On the other hand, due to the combinatorial generation of hashes, assuming symmetric density and fan-out for both database and sample hash generation, there are F times as many token combinations in the unknown sample to search for, and F times as many tokens in the database, thus the total speedup is a factor of about $1000000/F^2$, or about 10000, over token searches based on single constellation points.

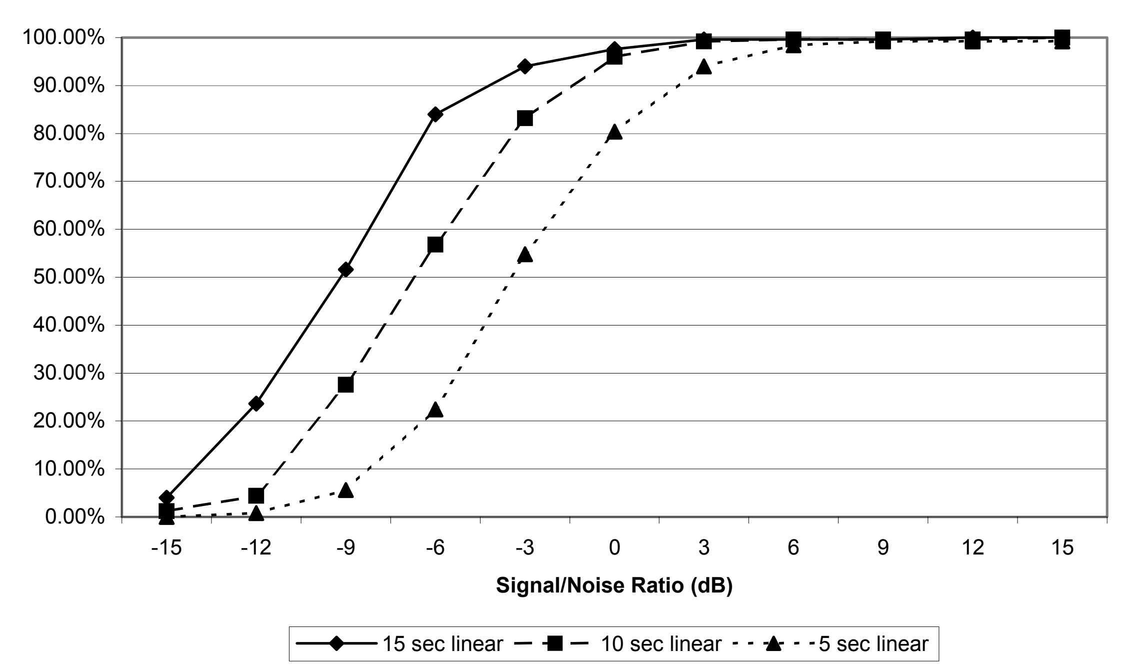
Note that the combinatorial hashing squares the probability of point survival, i.e. if p is the probability of a spectrogram peak surviving the journey from the original source material to the captured sample recording, then the probability of a hash from a pair of points surviving is approximately p^2. This reduction in hash survivability is a tradeoff against the tremendous amount of speedup provided. The reduced probability of individual hash survival is mitigated by the combinatorial generation of a greater number of hashes than original constellation points. For example, if F=10, then the probability of at least one hash surviving for a given anchor point would be the joint probability of the anchor point and at least one target point in its target zone surviving. If we simplistically assume IID probability p of survival for all points involved, then the probability of at least one hash surviving per anchor point is $p*[1-(1-p)^F]$. For reasonably large values of F, e.g. F>10, and reasonable values of p, e.g. p>0.1, we have approximately

$$p \approx p*[1-(1-p)^F]$$

so we are actually not much worse off than before.

We see that by using combinatorial hashing, we have traded off approximately 10 times the storage space for approximately 10000 times improvement in speed, and a small loss in probability of signal detection.

Figure 4: Recognition rate -- Additive Noise



Different fan-out and density factors may be chosen for different signal conditions. For relatively clean audio, e.g. for radio monitoring applications, F may be chosen to be modestly small and the density can also be chosen to be low, versus for the somewhat more challenging mobile phone consumer application. The difference in processing requirements can thus span many orders of magnitude.

2.3 Searching and Scoring

To perform a search, the above fingerprinting step is performed on a captured sample sound file to generate a set of hash:time offset records. Each hash from the sample is used to search in the database for matching hashes. For each matching hash found in the database, the corresponding offset times from the beginning of the sample and database files are associated into time pairs. The time pairs are distributed into bins according to the track ID associated with the matching database hash.

After all sample hashes have been used to search in the database to form matching time pairs, the bins are scanned for matches. Within each bin the set of time pairs represents a scatterplot of association between the sample and database sound files. If the files match, matching features should occur at similar relative offsets from the beginning of the file, i.e. a sequence of hashes in one file should also occur in the matching file with the same relative time sequence. The problem of deciding whether a match has been found reduces to detecting a significant cluster of points forming a diagonal line within the scatterplot. Various techniques could be used to perform the detection, for example a Hough transform or other robust regression technique. Such techniques are overly general, computationally expensive, and susceptible to outliers.

Due to the rigid constraints of the problem, the following technique solves the problem in approximately N*log(N) time, where N is the number of points appearing on the scatterplot. For the purposes of this discussion, we may assume that the slope of the diagonal line is 1.0. Then corresponding times of matching features between matching files have the relationship

$$t_k'=t_k+\text{offset},$$

where t_k' is the time coordinate of the feature in the matching (clean) database soundfile and t_k is the time coordinate of the corresponding feature in the sample soundfile to be identified. For each (t_k',t_k) coordinate in the scatterplot, we calculate

$$\delta t_k=t_k'-t_k.$$

Then we calculate a histogram of these δt_k values and scan for a peak. This may be done by sorting the set of δt_k values and quickly scanning for a cluster of values. The scatterplots are usually very sparse, due to the specificity of the hashes owing to the combinatorial method of generation as discussed above. Since the number of time pairs in each bin is small, the scanning process takes on the order of microseconds per bin, or less. The score of the match is the number of matching points in the histogram peak. The presence of a statistically significant cluster indicates a match. Figure 2A illustrates a scatterplot of database time versus sample time for a track that does not match the sample. There are a few chance associations, but no linear correspondence appears. Figure 3A shows a case where a

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase®
Smarter legal research.