Pergamon

Pattern Recognition, Vol. 30, No. 4, pp. 607–625, 1997
© 1997 Pattern Recognition Society. Published by Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0031–3203/97 $17.00+.00

PII: S0031-3203(96)00107-0

AUTOMATIC VIDEO INDEXING VIA OBJECT MOTION ANALYSIS

JONATHAN D. COURTNEY*
Texas Instruments, Incorporated 8330 LBJ Freeway, M/S 8374 Dallas, Texas 75243, U.S.A

(*Received* 12 *June* 1996; *received for publication* 30 *July* 1996)

Abstract—To assist human analysis of video data, a technique has been developed to perform automatic, content-based video indexing from object motion. Moving objects are detected in the video sequence using motion segmentation methods. By tracking individual objects through the segmented data, a symbolic representation of the video is generated in the form of a directed graph describing the objects and their movement. This graph is then annotated using a rule-based classification scheme to identify events of interest, e.g., appearance/disappearance, deposit/removal, entrance/exit, and motion/rest of objects. One may then use an index into the motion graph instead of the raw data to analyse the semantic content of the video. Application of this technique to surveillance video analysis is discussed. © 1997 Pattern Recognition Society. Published by Elsevier Science Ltd.

Video indexing   Object tracking   Motion analysis   Content-based retrieval

1. INTRODUCTION

Advances in multimedia technology, including commercial prospects for video-on-demand and digital library systems, have generated recent interest in content-based video analysis. Video data offers users of multimedia systems a wealth of information; however, it is not as readily manipulated as other data such as text. Raw video data has no immediate "handles" by which the multimedia system user may analyse its contents. By annotating video data with symbolic information describing the semantic content, one may facilitate analysis beyond simple serial playback.

To assist human analysis of video data, a technique has been developed to perform automatic, content-based video indexing from object motion. Moving objects are detected in the video sequence using motion segmentation methods. By tracking individual objects through the segmented data, a symbolic representation of the video is generated in the form of a directed graph describing the objects and their movement. This graph is then annotated using a rule-based classification scheme to identify events of interest, e.g., appearance/disappearance, deposit/removal, entrance/exit, and motion/rest of objects. One may then use an index into the motion graph instead of the raw data to analyse the semantic content of the video.

We have developed a system that demonstrates this indexing technique in assisted analysis of surveillance video data. The Automatic Video Indexing (AVI) system allows the user to select a video sequence of interest, play it forward or backward and stop at individual frames. Furthermore, the user may specify queries on video sequences and "jump" to events of interest to avoid tedious serial playback. For example, the user may select a person in a video sequence and specify the query "show me all objects that this person removed from the scene". In response, the AVI system assembles a set of video "clips" highlighting the query results. The user may select a clip of interest and proceed with further video analysis using queries or playback as before.

The remainder of this paper is organized as follows: Section 2 discusses content-based video analysis. Section 3 presents a video indexing technique based on object motion analysis. Section 4 describes a system which implements this video indexing technique for scene monitoring applications. Section 5 presents experimental results using the system. Section 6 concludes the paper.

* E-mail: courtney@csc.ti.com.

2. CONTENT-BASED VIDEO ANALYSIS

Video data poses unique problems for multimedia information systems that text does not. Textual data is a symbolic abstraction of the spoken word that is usually generated and structured by humans. Video, on the other hand, is a direct recording of visual information. In its raw and most common form, video data is subject to little human-imposed structure, and thus has no immediate "handles" by which the multimedia system user may analyse its contents.

For example, consider an online movie screenplay (textual data) and a digitized movie (video and audio data). If one were analysing the screenplay and interested in searching for instances of the word "horse" in the text, various text searching algorithms could be employed to locate every instance of this symbol as desired. Such analysis is common in online text databases. If, however, one were interested in searching for every scene in the digitized movie where a horse appeared, the task is much more difficult. Unless a human performs some sort of pre-processing of the video data, there are no symbolic keys on which to search. For a computer to assist in the search, it must analyse the *semantic* content of the video data itself. Without such capabilities, the information available to the multimedia system user is greatly reduced.

Thus, much research in video analysis focuses on semantic content-based search and retrieval techniques. Video *indexing* refers to the process of identifying important frames or objects in the video data for efficient playback. An indexed video sequence allows a user not only to play the sequence in the usual serial fashion, but also to "jump" to points of interest while it plays. A common indexing scheme is to employ *scene cut detection*(1) to determine breakpoints in the video data. Indexing has also been performed based on camera (i.e. viewpoint) motion(2) and object motion.(3,4)

Using breakpoints found via scene cut detection, other researchers have pursued *hierarchical segmentation*(5–7) to analyse the logical organization of video sequences. In the same way that text is organized into sentences, paragraphs, and chapters, the goal of these techniques is to determine a hierarchical grouping of video sub-sequences. Combining this structural information with *content abstractions* of segmented sub-sequences(8) provides multimedia system users a top–down view of video data.

The indexing technique described in this paper (the "AVI technique") performs video indexing based on object motion analysis. Unlike previous work, it forms semantically high-level interpretations of object actions and interactions from the object motion information. This allows multimedia system users to search for object-motion "events" in the video sequence (such as object entrance or exit) rather than features related to object velocity alone (such as "northeast movement").

3. VIDEO INDEXING VIA OBJECT MOTION ANALYSIS

Given a video sequence, the AVI technique analyses the motion of foreground objects in the data and indexes the objects to indicate the occurrence of several events of interest. It outputs a symbolic abstraction of the video content in the form of an annotated directed graph containing the indexed objects. This symbolic data may then be read by a user interface to perform content-based queries on the video data.

The AVI technique processes the video data in three stages: *motion segmentation*, *object tracking*, and *motion analysis*. First, motion segmentation methods(9,10) are used to segment moving foreground objects from the scene background in each frame. Next, each object is tracked through successive video frames, resulting in a graph describing object motion and path intersections. Then the motion graph is scanned for the occurrence of several events of interest. This is performed using a rule-based classifier which employs knowledge concerning object motion and the output of the previous stages to characterize the activity of the objects recorded in the graph. For example, a moving object that occludes another object results in a "disappear" event; a moving object that intersects and then removes a stationary object results in a "removal" event. An index is then created which identifies the location of each event in the video sequence.

Figure 1 depicts the relation between the video data, motion segmentation information, and the motion graph. Note that for each frame of the video, the AVI technique creates a corresponding symbolic "frame" to describe it.

3.1. *Terminology and notation*

The following is a description of some of the terms and notation used in the subsequent sections:

- A *sequence* $\mathcal{S}$ is an ordered set of N frames, denoted $\mathcal{S} = \{F_0, F_1, \ldots, F_{N-1}\}$, where F_n is frame number n in the sequence.
- A *clip* is a 4-tuple $\mathcal{C} = (\mathcal{S}, f, s, l)$, where $\mathcal{S}$ is a sequence with N frames, and f, s, and l are frame numbers such that $0 \leq f \leq s \leq l \leq N-1$. Here, F_f and F_l are the first and last valid frames in the clip, and F_s is the "start" frame. Thus, a clip specifies a sub-

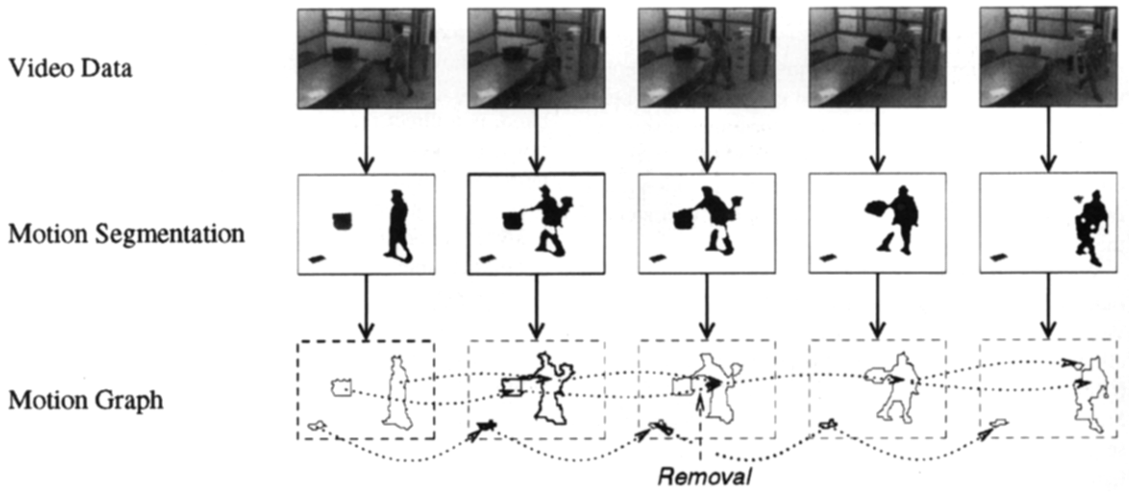

Fig. 1. Relation between video data, motion segmentation information, and the symbolic motion graph.

sequence and contains a state variable to indicate a "frame of interest".

- A *frame F* is an *image I* annotated with a timestamp *t*. Thus, frame number n is denoted by the pair $F_n = (I_n, t_n)$.
- An *image I* is an $r \times c$ array of pixels. The notation $I(i, j)$ indicates the pixel at coordinates (row i, column j). For purposes of this discussion, a pixel is assumed to be an intensity value between 0 and 255.
- A *timestamp* records the date and time that an image was digitized.

3.2. Motion segmentation

For each frame F_n in the sequence, the motion segmentation stage computes segmented image C_n as

$$C_n = \text{ccomps}(T_h \bullet k),$$

where T_h is the binary image resulting from thresholding the absolute difference of images I_n and I_0 at h, $T_h \bullet k$ the morphological *close* operation(12) on T_h with structuring element k, and the function ccomps($\cdot$) performs connected components analysis,(11) resulting in a unique label for each connected region in image $T_h \bullet k$. The image T_h is defined as

$$T_h(i,j) = \begin{cases} 1 & \text{if } |I_n(i,j) - I_0(i,j)| \geq h, \\ 0 & \text{otherwise,} \end{cases}$$

for all pixels (i,j) in T_h.

Figure 2 shows an example of this process. Absolute differencing and thresholding [Fig. 2(c) and (d)] detect motion regions in the image. The morphological close operation shown in Fig. 2(e) joins together small regions into smoothly-shaped objects. Connected components analysis assigns each detected object a unique label, as shown in Fig. 2(f). Components smaller than a given size threshold are discarded. The result is C_n, the output of the motion segmentation stage.

The motion segmentation technique described here is best suited for video sequences containing object motion within an otherwise static scene, such as in surveillance and scene monitoring applications. Note that the technique uses a "reference image" for processing. This is nominally the first image from the sequence, I_0. For many applications, the assumption of an available reference image is not unreasonable; video capture is simply initiated from a fixed-viewpoint camera when there is limited motion in the scene. Following are some reasons why this assumption may fail in other applications:

1. Sudden lighting changes may render the reference frame invalid. However, techniques such as scene cut detection(1) may be used to detect such occurrences and indicate when a new reference image must be acquired.
2. Gradual lighting changes may cause the reference image to slowly grow "out of date" over long video sequences, particularly in outdoor scenes. Here, more sophisticated techniques involving cumulative differences of successive video frames(13) must be employed.
3. The viewpoint may change due to camera motion. In this case, camera motion compensation(14) must be used to offset the effect of an apparent moving background.
4. An object may be present in the reference frame and move during the sequence. This causes the motion segmentation process to incorrectly detect the background region exposed by the object as if it were a newly-appearing stationary object in the scene.

A straightforward solution to problem 4 is to apply a test to non-moving regions detected by the motion segmentation process based on the following observation: if

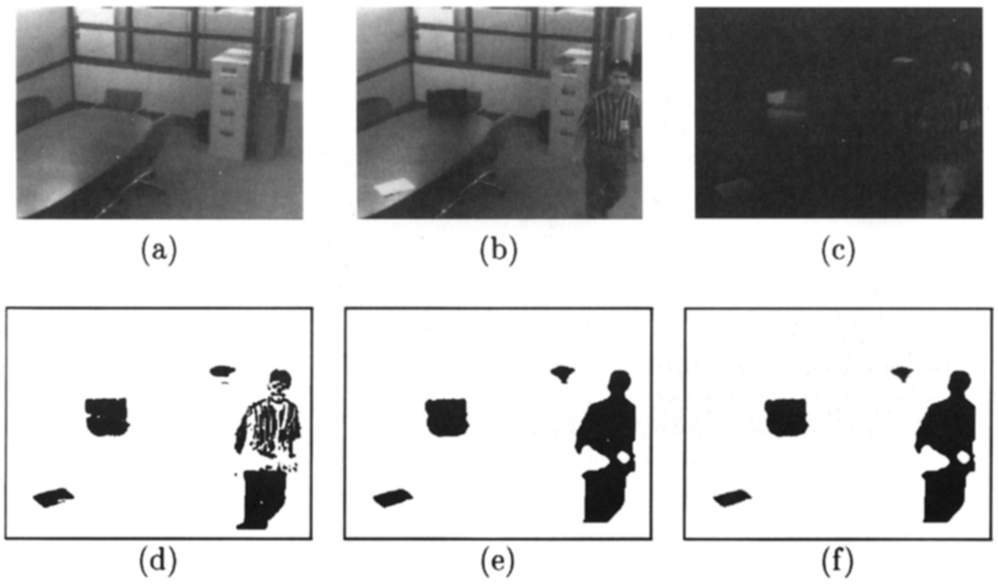

Fig. 2. Motion segmentation example. (a) Reference image I_0. (b) Image I_n. (c) Absolute difference $|I_n - I_0|$. (d) Thresholded image T_h. (e) Result of morphological close operation. (f) Result of connected components analysis.

the region detected by the segmentation of image I_n is due to the motion of an object present in the reference image (i.e. due to "exposed background"), a high probability exists that the boundary of the segmented region will coincide with intensity edges detected in I_0. If the region is due to the presence of a foreground object in the current image, a high probability exists that the region boundary will coincide with intensity edges in I_n. The test is implemented by applying an edge detection operator to the current and reference images and checking for coincident boundary pixels in the segmented region of C_n.[9] Figure 3 shows this process. If the test supports the hypothesis that the region in question is due to exposed background, the reference image is modified by replacing the object with its exposed background region (see Fig. 4).

No motion segmentation technique is perfect. The following are errors typical of many motion segmentation techniques:

1. True objects will disappear temporarily from the motion segmentation record. This occurs when there is insufficient contrast between an object and an occluded background region, or if an object is partially occluded by a "background" structure (for instance, a tree or pillar present in the scene).
2. False objects will appear temporarily in the motion segmentation record. This is caused by light fluctuations or shadows cast by moving objects.
3. Separate objects will temporarily join together. This typically occurs when two or more objects are in close proximity or when one object occludes another object.
4. Single objects will split into multiple regions. This occurs when a portion of an object has insufficient contrast with the background it occludes.

Instead of applying incremental improvements to relieve the shortcomings of motion segmentation, the AVI technique addresses these problems at a higher level where information about the semantic content of the video data is more readily available. The object tracking and motion analysis stages described in Sections 3.3 and 3.4 employ object trajectory estimates and knowledge concerning object motion and typical motion segmentation errors to construct a more accurate representation of the video content.

3.3. *Object tracking*

The motion segmentation output is processed by the object tracking stage. Given a segmented image C_n with P uniquely-labeled regions corresponding to foreground objects in the video, the system generates a set of features to represent each region. This set of features is named a "V-object" (video-object), denoted $V_n^p, p = 1, \ldots, P$. A V-object contains the label, centroid, bounding box, and shape mask of its corresponding region, as well as object velocity and trajectory information generated by the tracking process.

V-objects are then tracked through the segmented video sequence. Given segmented images C_n and C_{n+1} with V-objects $\mathbf{V}_n = \{V_n^p;\ p = 1, \ldots, P\}$ and $\mathbf{V}_{n+1} = \{V_{n+1}^q;\ q = 1, \ldots, Q\}$, respectively, the motion tracking process "links" V-objects V_n^p and V_{n+1}^q if their position and estimated velocity indicate that they correspond to the same real-world object appearing in frames F_n and F_{n+1}. This is determined using linear prediction of V-object positions and a "mutual nearest neighbor" criterion via the following procedure:

1. For each V-object $V_n^p \in \mathbf{V}_n$, predict its position in the next frame using
$$\hat{\boldsymbol{\mu}}_n^p = \boldsymbol{\mu}_n^p + \mathbf{v}_n^p \cdot (t_{n+1} - t_n),$$
where $\hat{\boldsymbol{\mu}}_n^p$ is the predicted centroid of V_n^p in C_{n+1}, $\boldsymbol{\mu}_n^p$ the centroid of V_n^p measured in C_n, $\mathbf{v}_n^p$ the estimated (forward) velocity of V_n^p, and t_{n+1} and t_n are the timestamps of frames F_{n+1} and F_n, respectively. Initially, the velocity estimate is set to $\mathbf{v}_n^p = (0, 0)$.
2. For each $V_n^p \in \mathbf{V}_n$, determine the V-object in the next frame with centroid nearest $\hat{\boldsymbol{\mu}}_n^p$. This "nearest neighbor" is denoted $\mathcal{N}_n^p$. Thus,
$$\mathcal{N}_n^p = V_{n+1}^r \ni \|\hat{\boldsymbol{\mu}}_n^p - \boldsymbol{\mu}_{n+1}^r\| \leq \|\hat{\boldsymbol{\mu}}_n^p - \boldsymbol{\mu}_{n+1}^q\| \quad \forall q \neq r.$$
3. For every pair $(V_n^p, \mathcal{N}_n^p = V_{n+1}^r)$ for which no other V-objects in $\mathbf{V}_n$ have V_{n+1}^r as a nearest neighbor, estimate $\mathbf{v}_{n+1}^r$, the (forward) velocity of V_{n+1}^r, as
$$\mathbf{v}_{n+1}^r = \frac{\boldsymbol{\mu}_{n+1}^r - \boldsymbol{\mu}_n^p}{t_{n+1} - t_n}; \qquad (1)$$
otherwise, set $\mathbf{v}_{n+1}^r = (0, 0)$.

These steps are performed for each C_n, $n = 0, 1, \ldots, N-2$. Steps 1 and 2 find nearest neighbors in the subsequent frame for each V-object. Step 3 generates velocity estimates for V-objects that can be unambiguously tracked; this information is used in step 1 to predict V-object positions for the next frame.

Next, steps 1–3 are repeated for the reverse sequence, i.e. $C_n, n = N-1, N-2, \ldots, 1$. This results in a new set of predicted centroids, velocity estimates, and nearest neighbors for each V-object in the reverse direction. Thus, the V-objects are tracked both forward and backward through the sequence. The remaining steps are then performed:

4. V-objects V_n^s and V_{n+1}^r are *mutual nearest neighbors* if $\mathcal{N}_n^s = V_{n+1}^r$ and $\mathcal{N}_{n+1}^r = V_n^s$. (Here, $\mathcal{N}_n^s$ is the nearest neighbor of V_n^s in the forward direction, and $\mathcal{N}_{n+1}^r$ is the nearest neighbor of V_{n+1}^r in the reverse direction.) For each pair of mutual nearest neighbors (V_n^s, V_{n+1}^r), create a *primary link* from V_n^s to V_{n+1}^r.
5. For each $V_n^p \in \mathbf{V}_n$ without a mutual nearest neighbor, create a *secondary link* from V_n^p to $\mathcal{N}_n^p$ if the predicted centroid $\hat{\boldsymbol{\mu}}_n^p$ is within ϵ of $\mathcal{N}_n^p$, where ϵ is some small distance.
6. For each V_{n+1}^q in $\mathbf{V}_{n+1}$ without a mutual nearest neighbor, create a *secondary* link from $\mathcal{N}_{n+1}^q$ to V_{n+1}^q if the predicted centroid $\hat{\boldsymbol{\mu}}_{n+1}^q$ is within ϵ of $\mathcal{N}_{n+1}^q$.

The object tracking procedure uses the mutual nearest neighbor criterion (step 4) to estimate frame-to-frame V-

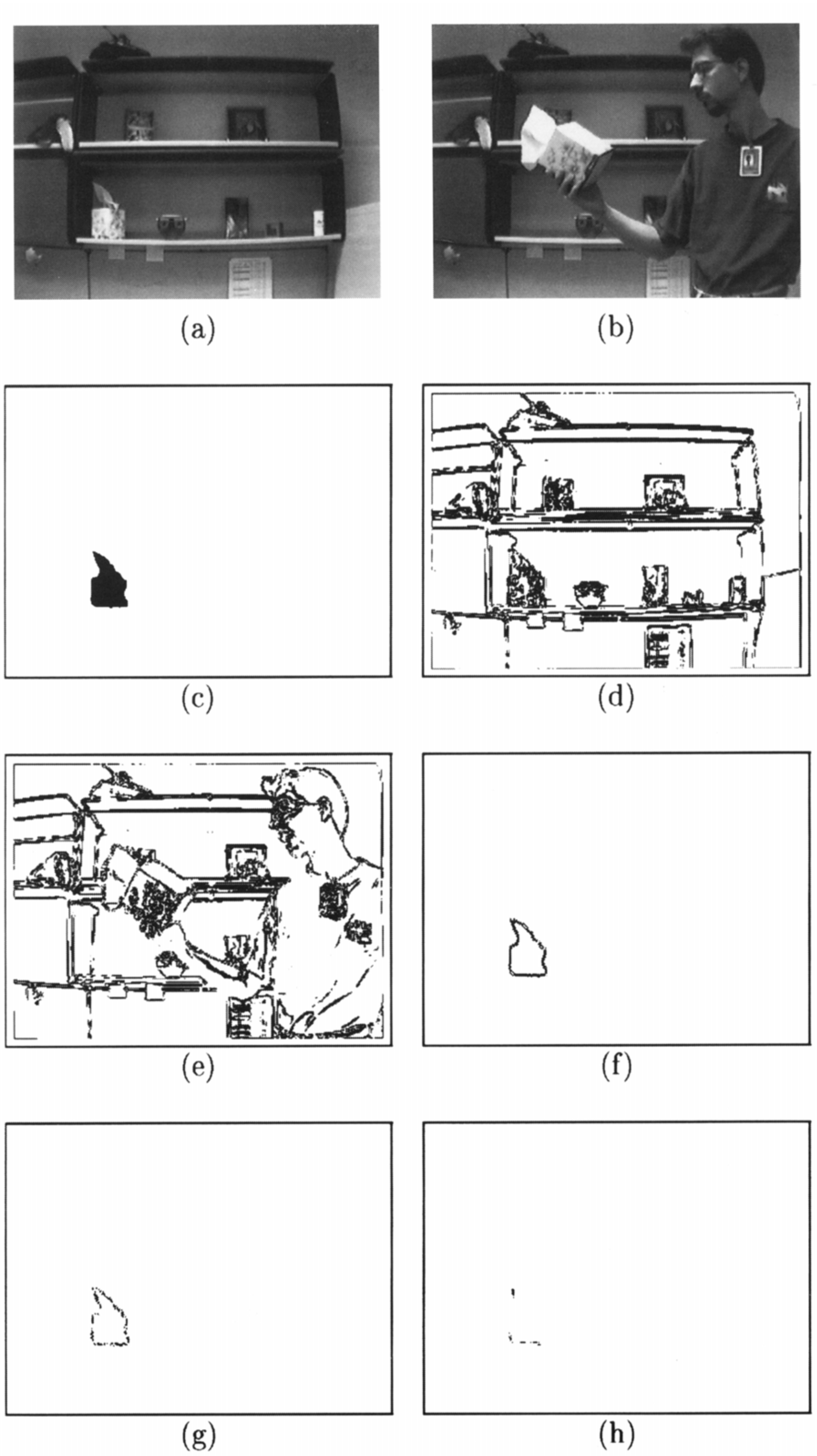

Fig. 3. Exposed background detection. (a) Reference image I_0. (b) Image I_n. (c) Region to be tested. (d) Edge image of (a), found using Sobel[11] operator. (e) Edge image of (b). (f) Edge image of (c), showing boundary pixels. (g) Pixels coincident in (d) and (f). (h) Pixels coincident in (e) and (f). The greater number of coincident pixels in (g) versus (h) support the hypothesis that the region in question is due to exposed background.

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.