Technical Disclosure Commons

Defensive Publications Series

April 28, 2015

CACHING IMAGE THUMBNAILS AS COMPRESSED VIDEO BITSTREAMS

Andrew Lewis

CACHING IMAGE THUMBNAILS AS COMPRESSED VIDEO BITSTREAMS

ABSTRACT

An image thumbnail caching system combines image thumbnails into one or more sets of video frames. The system then encodes the one or more sets of video frames to generate corresponding encoded video bitstreams. Then, the system stores the encoded video bitstreams. The system sends metadata associated with the encoded video bitstreams. Further, a specific thumbnail can be retrieved, for preview at a device, by decoding one or more of the encoded video bitstreams based on the metadata.

PROBLEM STATEMENT

Mobile applications often display long lists of image thumbnails representing large visual data sets, e.g., photo thumbnail galleries, video thumbnail galleries, page preview thumbnails. The image thumbnails are presented in scrolling list views that a user can navigate through. To display an image thumbnail on a device, the mobile application may have to store a corresponding image in the device's RAM, which is generally limited in the device. Once the image thumbnail is rendered, the application can store the image thumbnail in the cache of the device to reduce the time to display the image thumbnail. However, using an image encoder and decoder for storing and retrieving image thumbnails from a cache takes substantial time, memory, and power. A more advanced and efficient system for storing and retrieving image thumbnails from a cache to reduce time, memory, and power consumption is described.

DETAILED DESCRIPTION

The systems and techniques described in this disclosure relate to an image thumbnail caching system. The system can be implemented for use in an Internet, an intranet, or another client and server environment. The system can be implemented locally on a client device or implemented across a client device and server environment. The client device can be any electronic device such as a mobile device, a laptop, a smartphone, a tablet, a handheld electronic device, a wearable device etc.

Fig. 1 illustrates an example method 100 for transforming images into encoded video bitstreams. The method can be performed by a system that caches image thumbnails as video bitstreams, for example, the image thumbnail caching system.

The system combines image thumbnails into one or more sets of video frames (102). The system can combine multiple image thumbnails into a single video frame. The system can use various methodologies to combine image thumbnails. In one implementation, the system combines image thumbnails that the system expects to be displayed together on a device into a single video frame. For example, the system may choose to combine image thumbnails for images taken at a particular event or location or time into a single video frame. Alternatively, or additionally, the system combines the image thumbnails which the systems expects to be accessed together into a single video frame. For example, image thumbnails for pages of a book can be combined together into a single video frame. Once the system fills a video frame with image thumbnails, the system can tile a next set of image thumbnails into another video frame. The combined image thumbnails may be displayed close to each other in a listview, gridview, thumbnail view, or any other suitable format.

Further, the system encodes the one or more sets of video frames to generate corresponding encoded video bitstreams (104). The system can use a video encoder to encode video bitstreams from the one or more sets of video frames. The system may choose to employ any available video encoder to encode the video bitstreams. The system may select the video encoder based on various criteria, such as, number of video frames to be encoded, length of frames, throughput requirements, etc. In an example, a video encoder that is designed to handle a minimum of 30 frames per second is used to encode video bitstreams in a low power and high throughput environment. In an embodiment, the system can set a keyframe interval of the bitstream to the minimum. The keyframe interval can be made minimum by making every frame a key frame, thereby improving encoding or decoding speed. Keyframes are accessible without reference to other frames, thereby providing fast random access.

In an embodiment, the encoder can be configured to avoid using prediction structures that reference data for one thumbnail's region from another thumbnail's region, to help eliminate lossy compression artifacts that are correlated with the content of other thumbnails and to reduce distortion in stored thumbnails. For example, the top-left corner of a tiled thumbnail rectangle can be aligned to a macroblock boundary and its boundary macroblocks can be made not to predict its contents based on the macroblocks outside the thumbnail.

The system then stores the encoded video bitstreams in a storage or a memory associated with the device (106). The memory can be a built-in memory of the device or a remote memory at a server. Further, the system sends metadata associated with the stored encoded video bitstreams (108). The metadata can include identification information associated with each image

thumbnail, e.g., EXIF data, or frame number and/or location information of the thumbnail in the video. The system can send the metadata out-of-band.

Fig. 2 illustrates an example method 200 for retrieving images from the encoded video bitstreams. The system receives a request to preview a specific image thumbnail (202). The system can receive the request from a user of a device. The user may request to view a specific thumbnail from a photo or video thumbnail gallery, a page preview thumbnail for an electronic book, etc. In an example, the system receives a request to preview multiple image thumbnails together. The request includes a unique identifier for the required thumbnail.

On receiving the request, the system identifies the encoded bitstream that contains the specific image thumbnail (204). A mapping exists between thumbnail identifiers and encoded bitstreams. The system looks up the requested thumbnail identifier with the metadata associated with the encoded bitstreams to identify which encoded video bitstream that contains the requested image thumbnail.

Further, the system decodes the video frame that contains the specific image thumbnail from the identified encoded video bitstream (206). The system can use a video decoder to decode the video frame that contains the specific image thumbnail. The system may choose to employ any available video decoder to decode the video bitstreams. The system may select the video decoder based on various criteria, such as the number of video frames to be decoded, length of frames, throughput requirements, etc. In an example, a video decoder that is designed to handle a minimum of 30 frames per second is used to decode video bitstreams in a low power and high throughput environment. The video decoder can use metadata received from the video encoder to decode the video frame. The decoding can be performed at the device or at a server. Then, the

system provides the specific image thumbnail (208). The system can provide the specific image thumbnail to the user's electronic device for preview. The system may choose to employ any available decoder application program interface (API) to present the specific image thumbnail. In an example, if the system's applied decoder output surface is an Open Graphics Library (OpenGL) API texture, then sub-regions of the decoder's output surface texture can be shown in image views. Alternatively, or additionally, several OpenGL views in a scrolling list view may show the same output surface texture with different texture transformations applied to them.

Fig. 3 is a block diagram of an exemplary environment that shows components of a system for implementing the techniques described in this disclosure. The environment includes client devices 310, servers 330, and network 340. Network 340 connects client devices 310 to servers 330. Client device 310 is an electronic device. Client device 310 may be capable of requesting and receiving data/communications over network 340. Example client devices 310 are personal computers (e.g., laptops), mobile communication devices, (e.g. smartphones, tablet computing devices), set-top boxes, game-consoles, embedded systems, and other devices 310' that can send and receive data/communications over network 340. Client device 310 may execute an application, such as a web browser 312 or 314 or a native application 316. Web applications 313 and 315 may be displayed via a web browser 312 or 314. Server 330 may be a web server capable of sending, receiving and storing web pages 332. Web page(s) 332 may be stored on or accessible via server 330. Web page(s) 332 may be associated with web application 313 or 315 and accessed using a web browser, e.g., 312. When accessed, webpage(s) 332 may be transmitted and displayed on a client device, e.g., 310 or 310'. Resources 318 and 318' are resources available to the client device 310 and/or applications thereon, or server(s) 330 and/or

web pages(s) accessible therefrom, respectively. Resources 318' may be, for example, memory or storage resources; a text, image, video, audio, JavaScript, CSS, or other file or object; or other relevant resources. Network 340 may be any network or combination of networks that can carry data communication.

The subject matter described in this disclosure can be implemented in software and/or hardware (for example, computers, circuits, or processors). The subject matter can be implemented on a single device or across multiple devices (for example, a client device and a server device). Devices implementing the subject matter can be connected through a wired and/or wireless network. Such devices can receive inputs from a user (for example, from a mouse, keyboard, or touchscreen) and produce an output to a user (for example, through a display). Specific examples disclosed are provided for illustrative purposes and do not limit the scope of the disclosure.

DRAWINGS

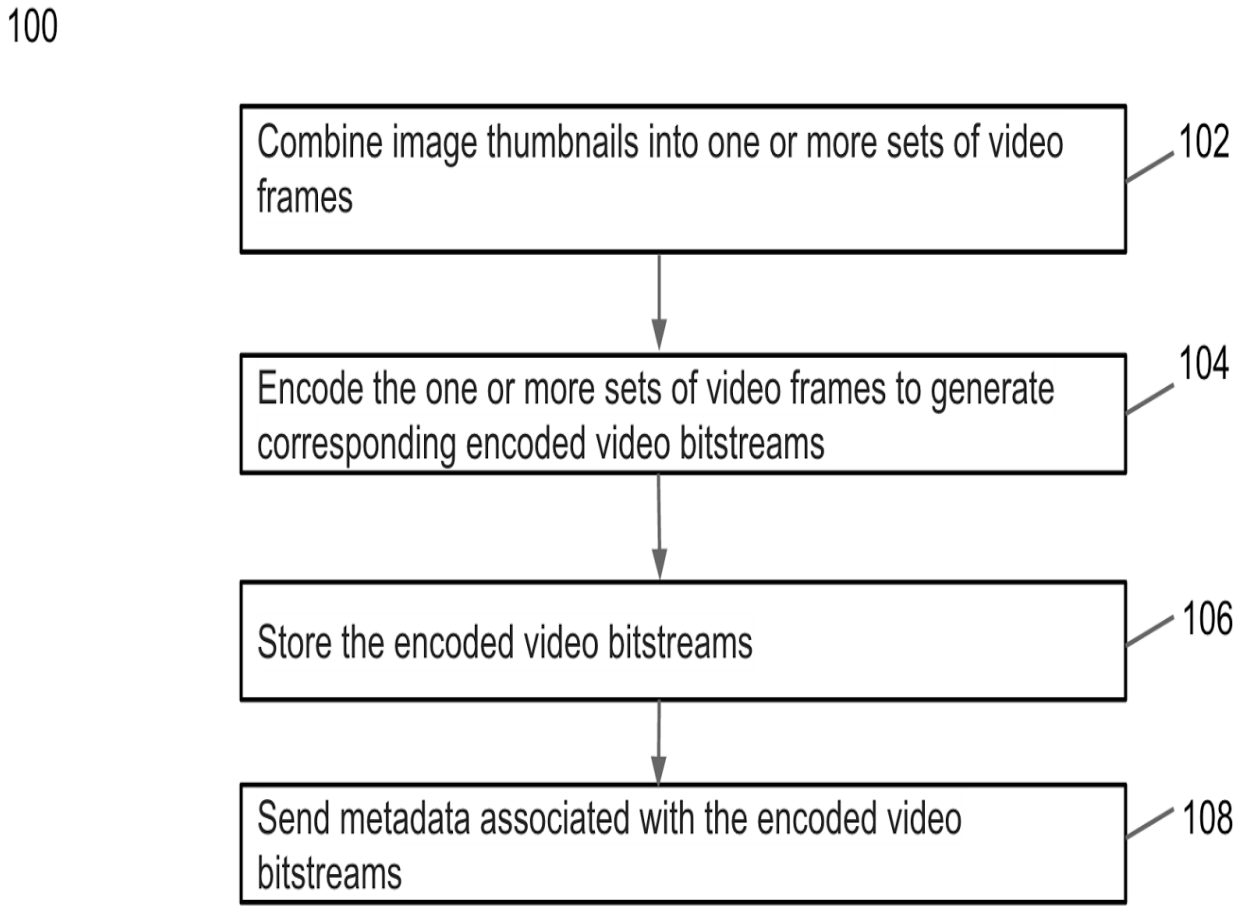

Fig. 1

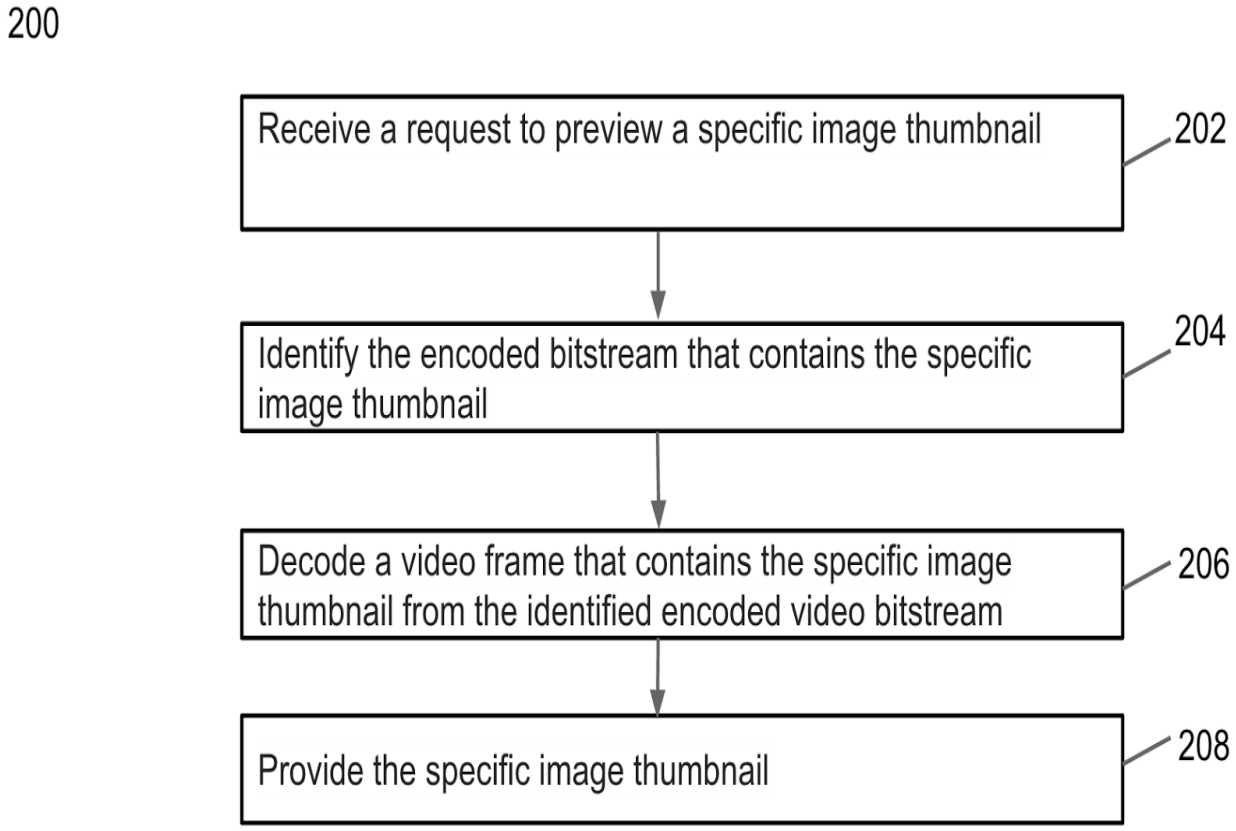
200
Receive a request to preview a specific image thumbnail
202
Identify the encoded bitstream that contains the specific image thumbnail
204
Decode a video frame that contains the specific image thumbnail from the identified encoded video bitstream
206
Provide the specific image thumbnail
208

Fig. 2

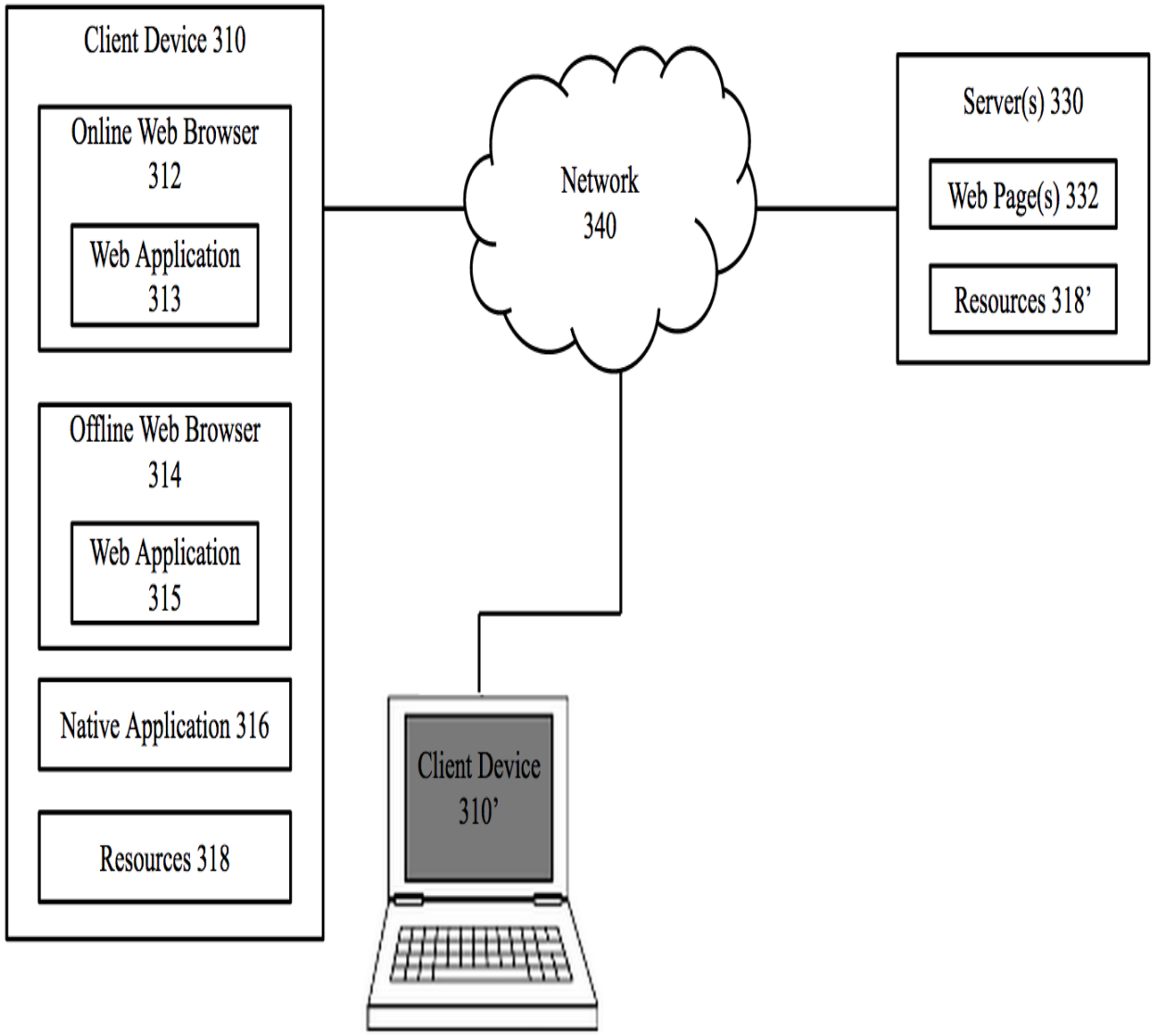
Client Device 310
Online Web Browser 312
Web Application 313
Offline Web Browser 314
Web Application 315
Native Application 316
Resources 318
Network 340
Server(s) 330
Web Page(s) 332
Resources 318'
Client Device 310'

Fig. 3