April 09, 2015

TESTING FOR CORRECTNESS AND REBASELINING OF IMAGES

Ravi Mistry

Elliot Poger

Stephen White

Brian Salomon

Robert Phillips

TESTING FOR CORRECTNESS AND REBASELINING OF IMAGES

ABSTRACT

An image correctness testing system compares an expected image with an image generated at a device. The system transmits the expected image to the device. The system then analyzes the image that is generated at the device in order to determine a checksum of the image generated at the device. The system further compares the checksum of the image generated at the device with a checksum of the expected image and presents a difference to a user on a web interface. The system can then receive an input from the user to rebaseline the image generated at the device and/or the expected image. On receiving the user input, the system updates the checksum of the expected image equal to the checksum of the image generated at the device or vice versa.

PROBLEM STATEMENT

Browsers and graphics libraries validate their repository checkins by comparing images generated at a client device with change(s) against expected images. When many different combinations of platforms, hardware and operating systems need to be tested, each combination could require its own sets of images, resulting in a large set of images (millions of images). Hence, testing for correctness of millions of images against expected images can become a time consuming and tedious process. A method and system that provides a framework for efficiently testing the correctness of a large set of images is described.

IMAGE CORRECTNESS TESTING SYSTEM

The system and techniques described in this disclosure relate to an image correctness testing system that compares a checksum of an expected image with a checksum of an image generated at a device. The image correctness testing system can be implemented for use in an Internet, an intranet, or another client and server environment. The image correctness testing system can be program instructions 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, laptop, a smartphone, a tablet, or a handheld electronic device, etc.

Fig. 1 illustrates an example method 100 for testing correctness of images by comparing a checksum of an expected image with a checksum of an image generated at a device. Method 100 can be performed by a system that tests correctness of images using the checksum, e.g., the image correctness testing system.

As shown in Fig. 1, the system transmits an expected image (110) to a device. The expected image can be stored on a memory associated with the system, on a cloud storage, etc. The system may transmit the expected image to the device using Internet, intranet, short range communication protocols, e.g., bluetooth or WiFi, or any other client and server environment. The device can be any electronic device such as a laptop, mobile device, a tablet, or a handheld electronic device. In another implementation, the system can transmit the expected image from a memory to a display of the same device.

The system analyzes the image generated at the device (120). During the transmission from the system memory to the device or while processing the image at the device, the expected image may undergo some changes. For example, the expected image may get distorted because

of noise in a transmission channel or the expected image may be modified by the device based on the operating system of the device. The image generated on a display of the device can thus be different from the expected image. The system then analyzes the image generated at the device to determine a checksum of the image. The checksum is a small-size datum or numerical value from an arbitrary block of digital data that is used to detect errors introduced during generation, transmission, or storage. The checksum acts as a signature for the digital data because the numeric value of the checksum gets modified if the digital data is changed.

On determining the checksum, the system compares the checksum of the image generated at the device with the checksum of the expected image (130). If the image generated at the device and the expected image are identical, the checksum remains the same. However, if the image generated at the device is different from the expected image, the system calculates a difference between the image generated on the device and the expected image based on the checksum. The system can calculate the difference in terms of percentage change, degree of variation, etc.

The system further presents the difference between the image generated at the device and the expected image to a user (140). The system presents the expected image, image generated on the device, and the difference to the user in a web interface. Alternatively or additionally, the web interface may also present a difference between the expected image and other images as received on various devices with different operating systems, platforms, and hardwares. In certain scenarios, when the system handles large set of images, the system displays only a subset of images to the user. Images may be sorted according to different algorithms, e.g., differing pixels in white, perceptual differences, the top X, or a configurable number. Further, the system

also allows the user to narrow down the set of images based on platforms, operating systems, hardware, test name, image name, etc via the web interface.

The system further receives an input from the user to rebaseline the expected image and/or the image generated at the device (150). While analysing the difference, the user may decide to rebaseline the expected image and/or the image generated at the device. Rebaselining an image refers to updating the pixels, text baselines, or checksum, etc., of the image. In an example scenario, while analyzing the difference, the user may consider the image generated at the device to be better than the expected image, hence the user can decide to rebaseline the expected image into the image generated at the device. The web interface also provides an option for the user to rebaseline multiple images at a same time.

On receiving the user input, the system updates the checksum of the expected image equal to the checksum of the image generated at the device or vice versa (160) based on the user input. By updating the checksum of the expected image equal to the image generated at the device, the system rebaselines the expected image into the image generated at the device.

Fig. 2 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 210, servers 230, and network 240. Network 240 connects client devices 210 to servers 230. Client device 210 is an electronic device. Client device 210 may be capable of requesting and receiving data/communications over network 240. Example client devices 210 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 210' that can send and receive data/communications over network 240. Client device 210 may execute

an application, such as a web browser 212 or 214 or a native application 216. Web applications 213 and 215 may be displayed via a web browser 212 or 214. Server 230 may be a web server capable of sending, receiving and storing web pages 232. Web page(s) 232 may be stored on or accessible via server 230. Web page(s) 232 may be associated with web application 213 or 215 and accessed using a web browser, e.g., 212. When accessed, webpage(s) 232 may be transmitted and displayed on a client device, e.g., 210 or 210'. Resources 218 and 218' are resources available to the client device 210 and/or applications thereon, or server(s) 230 and/or web pages(s) accessible therefrom, respectively. Resources 218' 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 240 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

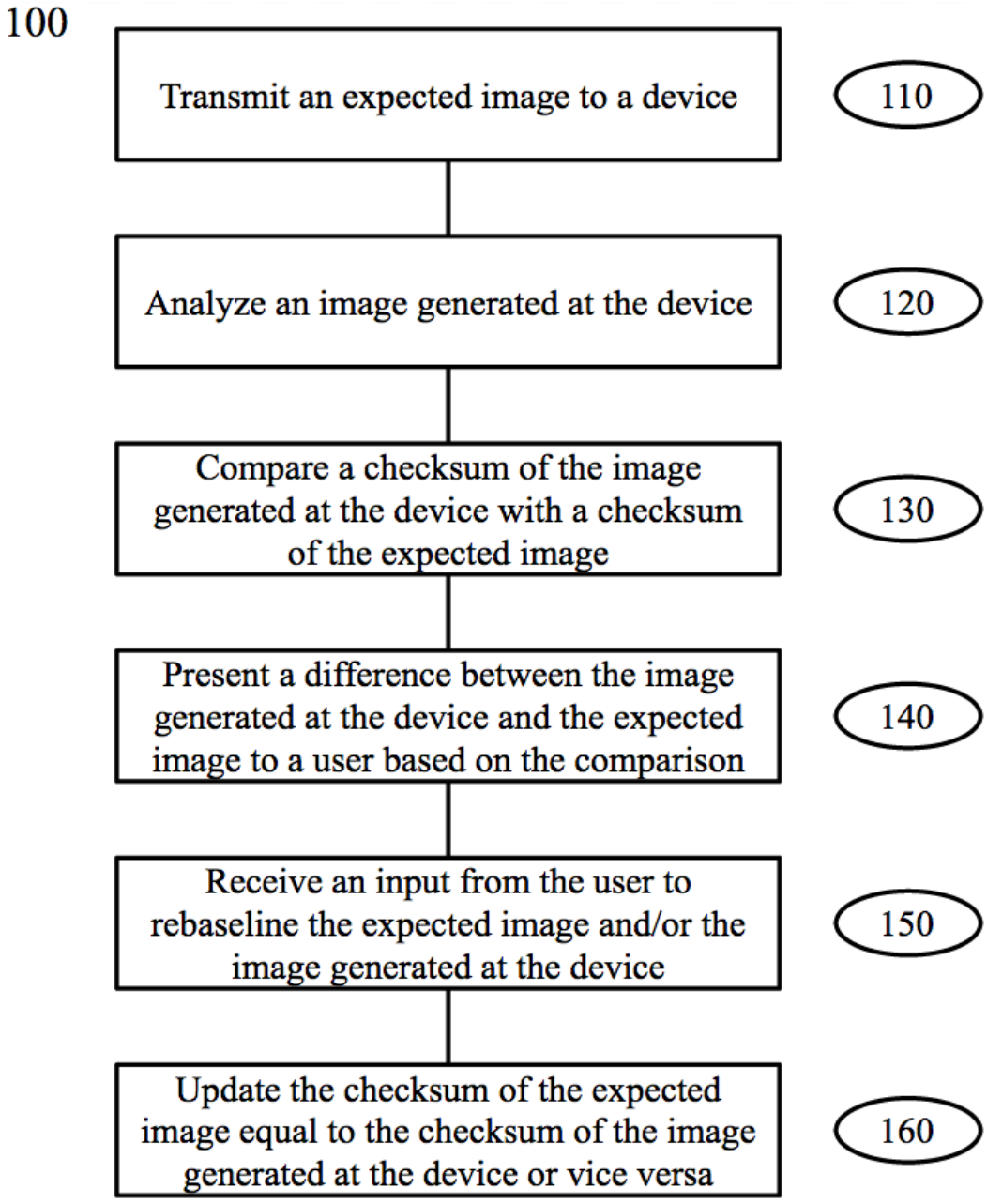

Fig. 1

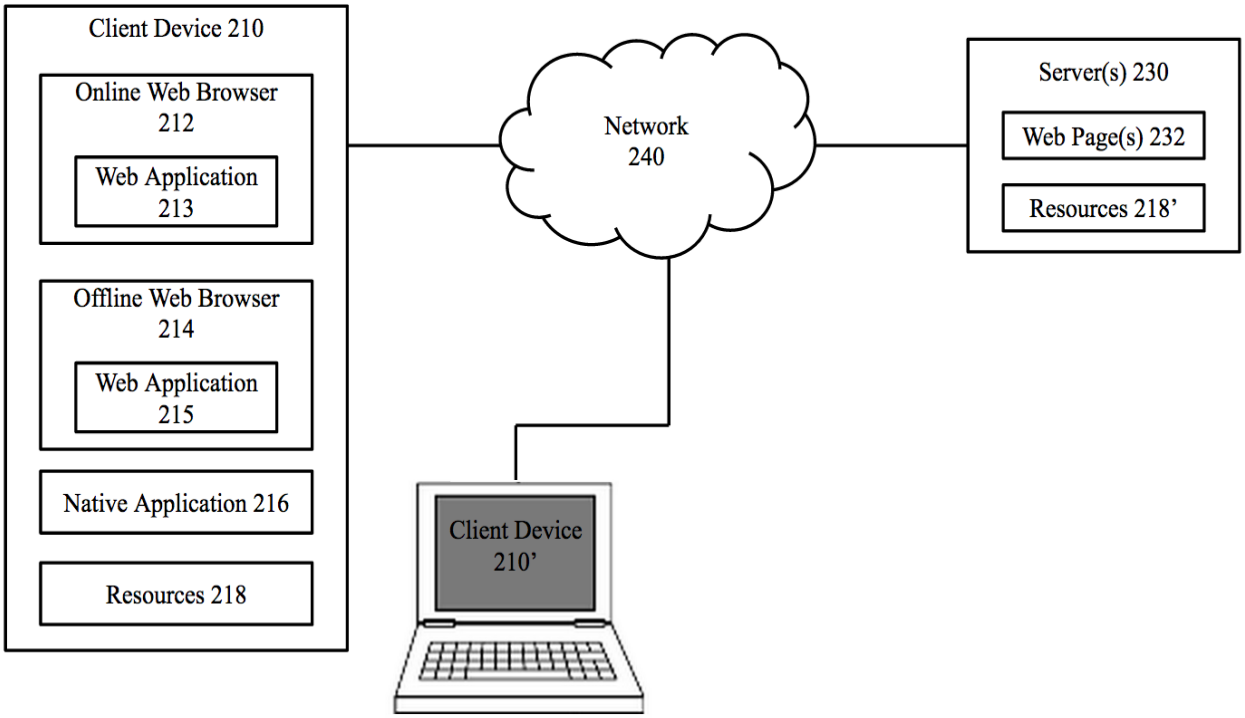
Client Device 210
Online Web Browser 212
Web Application 213
Offline Web Browser 214
Web Application 215
Native Application 216
Resources 218
Network 240
Server(s) 230
Web Page(s) 232
Resources 218'
Client Device 210'

Fig. 2