

Distributed Computing and Image Processing in JavaScript

A proof of concept realizing webcam motion detection in JavaScript.

This document describes a proof of concept that realizes motion detection for the webcam software MJPG-Streamer in JavaScript on a remote computer. The motivation for this project is the scope of the MJPG-Streamer server software to be installed on embedded devices. The server is running on embedded devices like routers, which are incapable of performing image processing quickly due to their limited resources. This can be solved by offloading the CPU intense task of motion detection to another computer. For a regular computer, the task of detecting motion in a webcam stream is feasible.

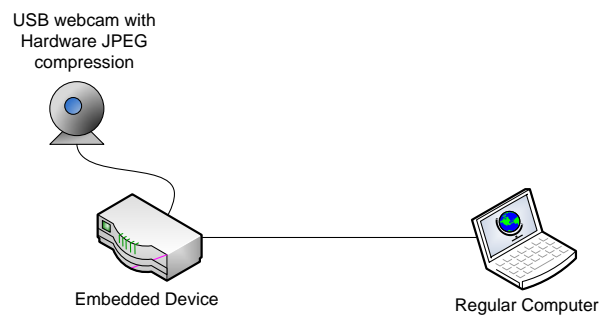


Figure 1: An overview of the setup for motion detection realized in JavaScript. The embedded device runs the server “MJPG-Streamer”; the regular computer has a high bandwidth connection to the embedded device.

Instead of developing proprietary client software for the regular computer, a browser is used as client software. In particular, the browser Firefox 3.0.7 is used for this proof of concept, because it supports the “canvas” object and offers a reasonable JavaScript execution performance.

The embedded device runs the server Software “MJPG-Streamer”. MJPG-Streamer takes images of a webcam (or other input plugins) and serves them to its output plugins. In this case, the output plugin is a very basic HTTP webserver. By using a browser it is possible to watch the live stream of pictures and to request single snapshots. MJPG-Streamer was developed because it can make use of the hardware

compression in a few compatible webcams¹, offloading the CPU intense task of compressing pictures to JPEG to the webcam hardware.

To start the motion detection, the browser on the computer is pointed to a JavaScript page served by the server software. The JavaScript page is rendered and the included script is executed. The script downloads current snapshots of the webcam and compares these snapshots for differences. If the previous and the most recent snapshot differ too much, an AJAX request is send to the server, informing him of the motion event. This way, the client performed the image analysis and the server just receives notification if motion was detected.

In order to compare pictures in JavaScript, the pictures are rendered to a canvas area with the JavaScript function “drawImage ()”. During this step, the image data is decoded and scaled as well by the browser. With the images rendered to the canvas drawing area, pixel data can be read back with the function “getImageData ()”. After this step of reading back the data, the pixels red, green, blue and alpha values are stored in an array. By comparing the values of the most recent and the previous picture, differences are counted and drawn to the lower part of the canvas area. If enough pixels changed from one snapshot to the other, this is interpreted as motion and an AJAX request is send to the server. The MJPG-streamer currently just registers this event and prints a message to the console it was started from. Implementing actions on the event of motion are out of scope of this proof of concept and can follow later.

For this proof of concept, just one client with a browser is needed and used. Scaling the task to several clients and reducing the workload for each client should be done at a later stage. Also, currently the server has to trust the results of the computing client, it is necessary to find solutions that allow validating submitted results.

¹ The author is using a Logitech Quickcam Sphere AF, a Logitech Pro 5000 was successfully tested as well.

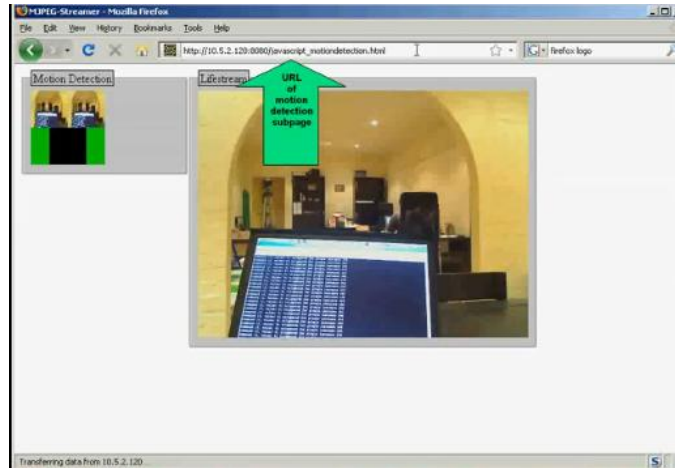


Figure 2: Screenshot of the published video. The large screen shows the live-stream, the small area is used for the motion detection.

The proof of concept shows, that a web-browser can be used as computing client and that modern web browsers can do image processing like motion detection. A short video of the test² is available at YouTube.

In the near future, the MJPG-streamer author intends to make the autofocus plugin obsolete by this technique. Instead of manually adjusting the focus or performing the focus calculation on the server side, it is possible to optimize the sharpness and send commands to the webcams³ focus adjustment.

The sourcecode of MJPG-Streamer and of the motion detection webpage⁴ is available at the project website⁵.

² http://www.youtube.com/watch?v=u3_cFel26J8

³ Known to work with a Logitech QC Sphere AF and Pro 9000

⁴ http://mjpg-streamer.svn.sourceforge.net/.../www/javascript_motiondetection.html

⁵ <http://mjpg-streamer.sf.net>