Automated image capturing + API

Description	Image capture + image & video generation + API
Period	Fall 2014 – Ongoing
Languages & Libs	Bash, PHP
Tags	GitHub
GitHub	nikonyrh/webcammon

Out of interest on nature observation, computer vision, image processing and so forth I developed an automated system to capture one photo / minute and store it on a disk. The project also has Bash and PHP scripts coordinating external tools such as montage for image stitching and mencoder for video generation. PHP also provides an HTTP API for image generation and file size statistics.

Images are captured by a cronjob calling grabFrame.sh once a minute. It configures the webcam before each shot to use fixed parameters for parameters such as exposure, contrast and color temperature to quarantee more constant outcome. File name is based on the current UTC timestamp to avoid problems when transitioning in and out from the daylight saving time. Configurationis handled by calling V4l2-ctl and capture is handled by fswebcam. Images are 30 - 400 kb so one day's 1440 photos take about 350 Mb, depending on the season. Example photos from 7 am can be seen in Figure 1.

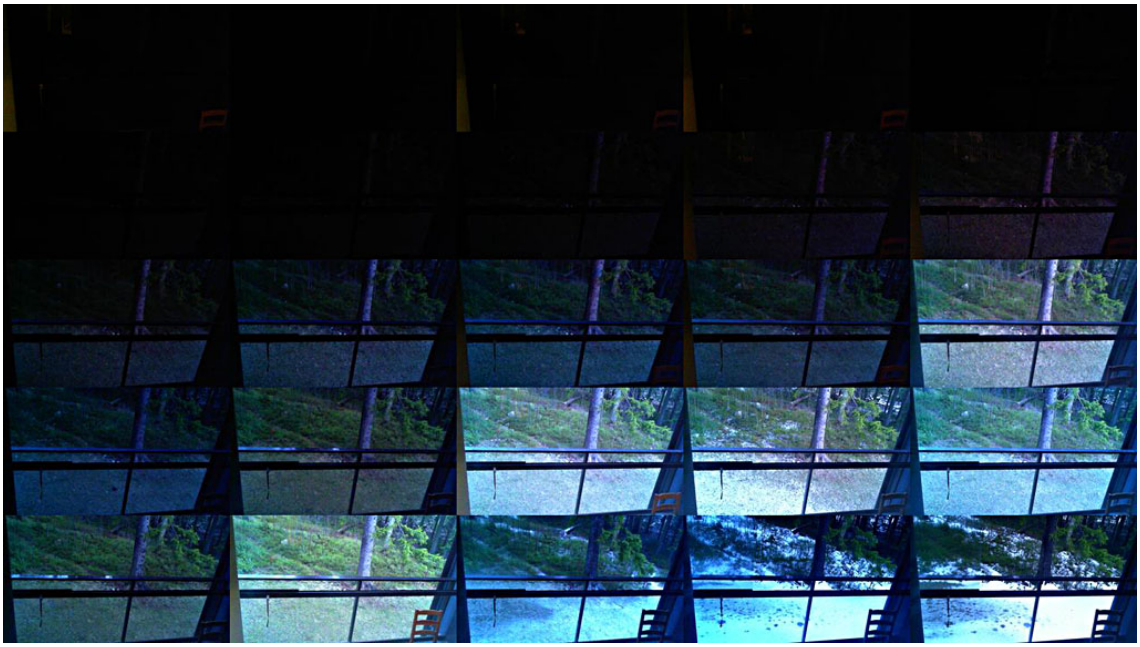

Figure 1: Morning 7 o'clock on 15 consecutive days in March shows the lengthening of day light duration.

Figure 1 was generated by creating a list of Unix epoch timestamps which indicate that at which points of time images should be shown. Then existing images are iterated over and for each timestamp the most closely matching image is remembered. The list of file names is then passed on montage command-line software which produces the final JPG which is cached on the server. All this happens by simply calling the URL **/imageGen/07.jpg?grid=5** and it tiles last 25 (5×5) images which were captured at 7am.

An other time-based API call looks like **/imageGen/1d.jpg?w=1280&grid=8&skip=30w**. It generates a tiled 8×8 images throughout one day 30 weeks ago, resulting in 22.5 minutes / image. It should be visible in GitHub at nikonyrh/webcammon/master/samples/1d.jpg.

Once images had been collected for a few days I got interested on what kind of analysis and visualization could be done from them. An easy option was to just observe JPG files sizes and group them based on the date and the time of day. Since the number of images will be in hundreds of thousands it was necessary to make this operation fast. If the day has passed then no new photos will bee added to that folder and thus file sizes can be saved on a "cache file" on disk. It is a lot faster to read a single small file than to iterate over 1440 file's metadata. The HTML based UI's visualization of file sizes is shown in Figure 2.

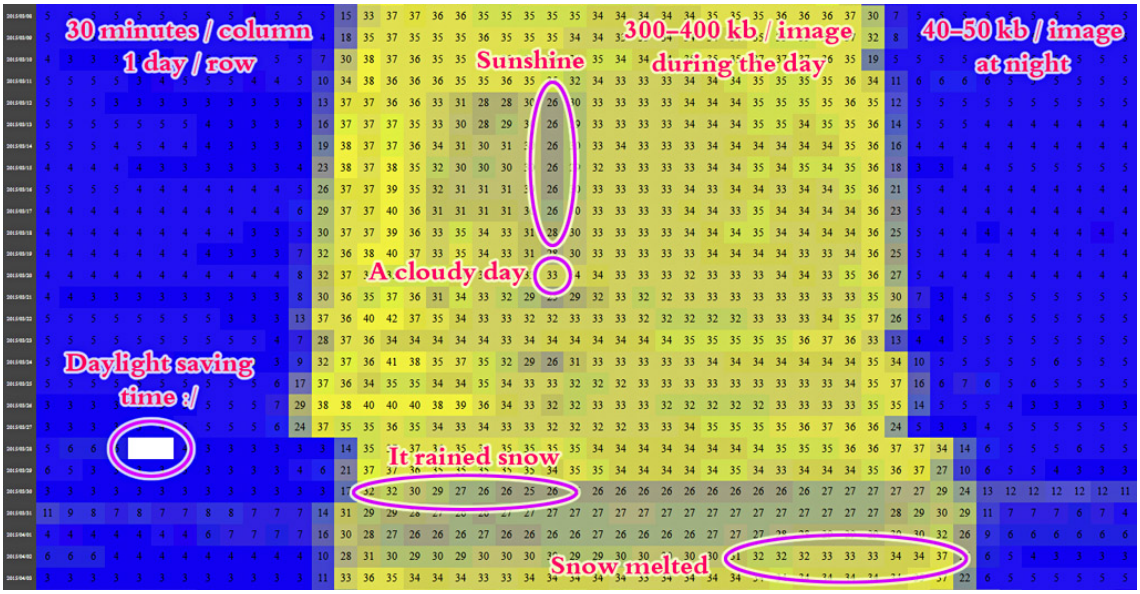

Figure 2: File sizes (x 10 kb) on different days and times of day at 30 minutes resolution. Weather effects such as bright sunshine and snowfall can clearly been detected by just observing file sizes.

The project also has an API for querying file size statistics in a plain text format which can be easily exported to external programs such as Matlab. An example output can be seen below. First three columns are the year, month and day and the rest are median file sizes for each hour of the day. Full rows are shown at GitHub.

```
  -1 -1 -1    0.00    1.00    2.00    3.00    4.00    5.00    6.00    7.00    8.00    9.00   10.00
2015 03 26   52.20   32.20   31.50   31.20   42.20   53.90  176.80  380.60  402.70  389.10  374.80
2015 03 27   32.30   32.10   31.60   33.90   53.20   53.00  147.60  357.50  351.30  341.90  335.60
2015 03 28   52.70   57.90   64.50   37.60   30.60   30.60   86.80  356.20  363.00  349.70  348.90
2015 03 29   51.90   30.50   30.70   30.70   30.50   36.80  132.20  370.60  358.90  349.40  348.40
2015 03 30   31.20   31.00   30.40   30.70   30.50   30.70  100.70  315.90  293.60  267.30  255.70
2015 03 31   96.40   77.30   74.40   74.30   76.30   72.00  225.10  290.10  276.70  262.60  266.30
2015 04 01   41.40   39.00   40.60   50.50   68.50   66.30  231.00  272.90  257.70  265.40  260.30
2015 04 02   60.40   49.30   37.60   38.00   37.80   38.60  187.20  303.00  294.80  296.50  299.40
2015 04 03   31.40   31.60   31.80   31.60   31.50   32.50  217.90  359.90  340.50  334.70  338.10
2015 04 04   53.50   53.90   53.80   43.70   30.70   31.80  182.00  366.70  350.10  343.20  346.20
```

The final currently implemented feature is the parallel rendeding of videos out of selected images, which is implemented in createVideo.sh. It starts by first listing all desired images' file names in the correct order and splitting them into four equal chunks. All of these chunks are then processed in parallel (by utilizing `"xargs -P0 -n1"`) and each produces a segment of the video. These segments are then concatenated together and temporary files are deleted. The actual video generation is done by mencoder, which has relatively simple command line options. One interesting future feature would be blending together images from different seasons together into a single high-res image instead of the currently implemented tiling.