Exercise 2
Daniel Lindberg
06/15/2018

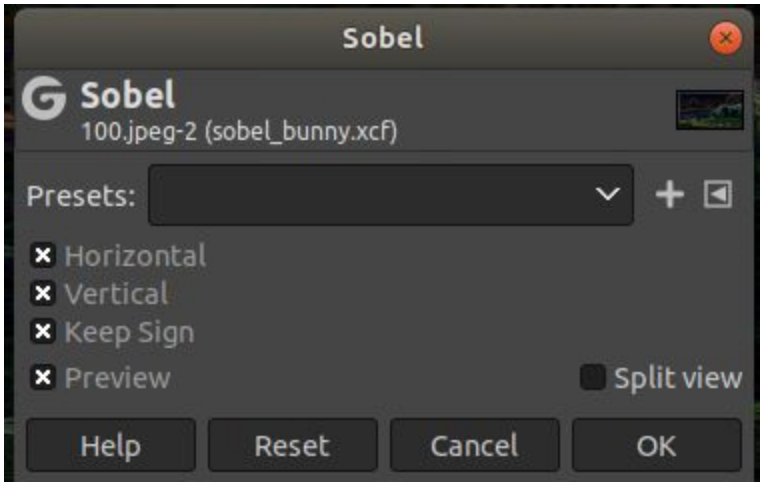
Step 1
Command I ran to obtain the ffmpeg: ffmpeg -i big_buck_bunny_480p_surround-fix.avi -r 1/1 $BunnyFrames%03d.jpeg

Here is 100th frame:

You can also view it in my Step 1 folder by selecting 100.jpeg.

Step 2
Attached is my sobel filter applied to the 100th frame:

You can also find it in the Step 2 folder, named as: sobel_bunny.jpg
The settings I used for the sobel filter in gimp were:



Step 3

Capture Viewer

Capture Viewer uses Linux's Video for Linux API. This API offers a number of tools to help stream video. V4L2 checks the image formats and colorspace that your webcam supports and provide. V4l2_format structure is mean to be changed to match your camera's format. It then requests buffers to allocate space for device buffers for Streaming I/O. Finally there is also query buffers that help get in raw data. Using the function provided here:

```
if(-1 == xioctl(fd, VIDIOC_QUERYBUF, &buf))
{
    perror("Querying Buffer");
```

```
    return 1;
}
```

The mmap function maps length bytes starting at the offset provided in the device. Next in the code provided they capture the image using the following commands in the main loop:

```
fd_set fds;
        struct timeval tv;
        int r;

        FD_ZERO(&fds);
        FD_SET(fd, &fds);

        /* Timeout. */
        tv.tv_sec = 2;
        tv.tv_usec = 0;

        r = select(fd + 1, &fds, NULL, NULL, &tv);

        if (-1 == r)
        {
                if (EINTR == errno)
                continue;
                errno_exit("select");
        }
```

These commands take the frame and save it into the buffer. It is then streamed.

Simple-Capture

This code , like capture-viewer leverages the Video for Linux's API.

This code is a little unique in the fact that it process the frames of the image. If you go into the process_image function you investigate that it has a yuv2rgb function. YUV is a similar color scheme to RGB. However YUV colored images typically take the weighted values of red, green and blue are summed to produce a Y', which is a measure of overall brightness. Then U and V are computed as scaled differences between Y' and the B and R values. This Y' value is meant to account for human perception, allowing for reduced bandwidth for the signal used in video camers to display color. This YUV system seems to be used so it can produce a signal suitable for reception on old monochrome displays. Since the human eye has fairly little spatial sensitivty to color, the accuracy of the brightness information of the luminance (Y') has more impact on the image detail discerned.

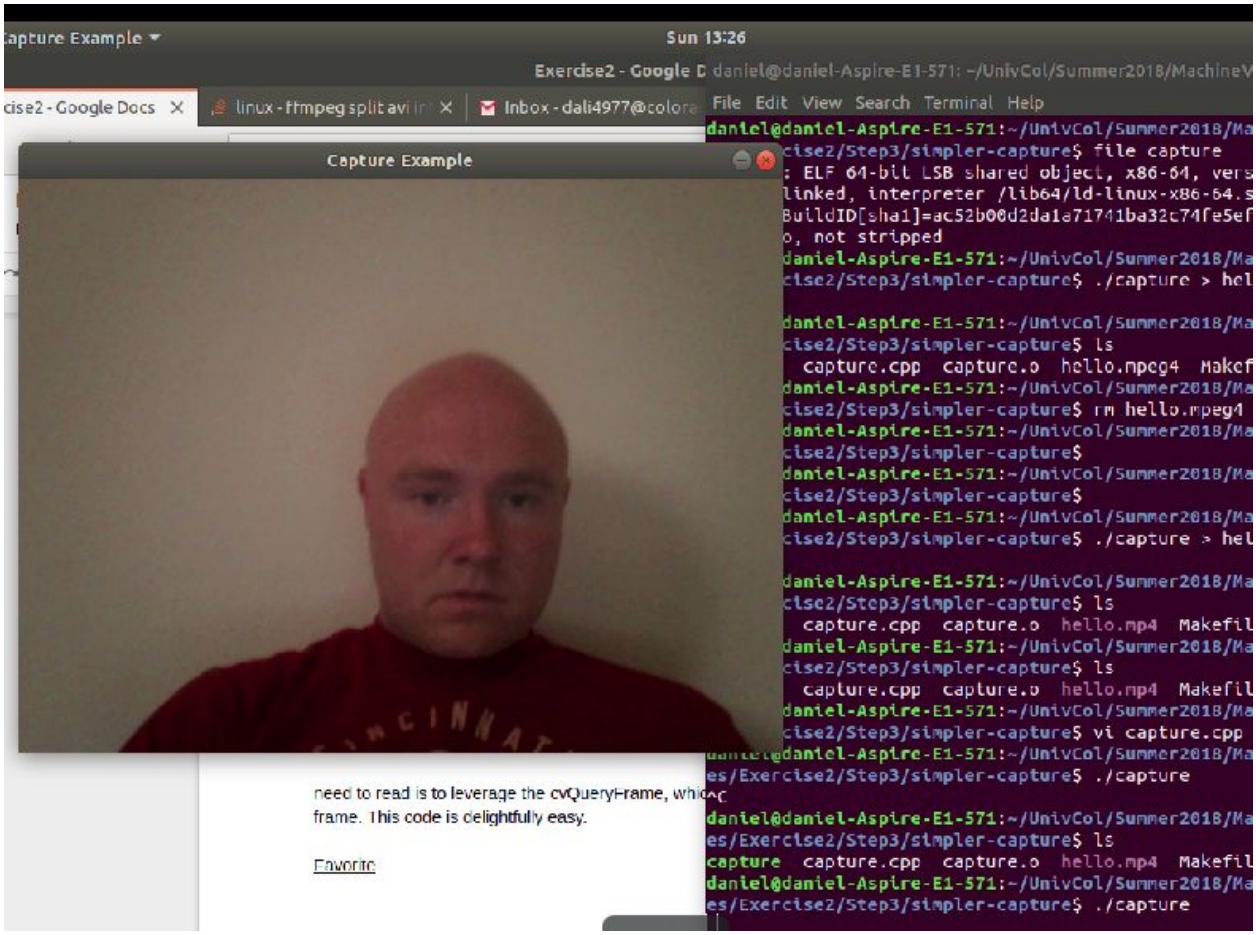
Simpler-Capture

Simpler capture is delightfully simple. It leverages OpenCV's API. It uses the cvCreateCameraCapture(0) function to obtain the camera feed. Then simply to read it, all you

need to read is to leverage the cvQueryFrame, which keeps an active pointer to the most recent frame. This code is delightfully easy.

Favorite

You can go to the step 3 folder and within it is an image labelled: FavoriteFrame.png. That is a screenshot of my favorite code. My favorite is simpler-capture, I like how simple the code is to grab the stream from the camera display. OpenCV library seems like it has an ability to leverage difficult tasks in terms of Machine vision and built simple functions to access them. Screenshot below:



Attached in the Step 3 folder, specifically within the simpler-capture you can find the changes I wrote for the code. I commented it as well. It gets the timeframe of each frame, it computers the difference between these frames. It computes the jitter, average case and the worst case. To run it you have to do a ‘make’ followed by a ‘./capture’.

Step 4

For step 4 I wrote a python version of the code. It is within the step 4 folder, then the python-transformer code. To run it simply run “python capture-transformer.py”. When it is ran it starts out as displaying canny. To run the sobel code, type ‘s’ or ‘S’ and then hit enter within the

terminal. To return back to canny, hit the ‘c’ or ‘C’ and then enter again. To stop the program, hit the ‘q’ button then hit enter in the terminal.

Important Notes

-Below are the outputs for about a minute of canny and about a minute of sobel. Sobel has a slightly faster framerate than canny. However canny has a faster worst case than the sobel. I didn’t print all of the frame information for a minute because it would make the report a little longer.

- The Canny frame size I used was set to have a threshold between 100 and 200. You can change this on line 35
- The Sobel frame will take the frame and display the gradient in the x and y position. The kernel size is set to 3 and we use 64F datatype defined by open cv

Canny information:

Frame:127 jitter:0.00545938607234
Frame:128 jitter:-0.00291745070439
Frame:129 jitter:-0.00269572142583
Frame:130 jitter:-0.00306741599065
Frame:131 jitter:-0.00261060599309
Frame:132 jitter:-0.00300161246282
Frame:133 jitter:-0.00294939879399
Frame:134 jitter:-0.00270954970342
Frame:135 jitter:-0.00385658148747
Frame:136 jitter:-0.00179259184819
Frame:137 jitter:-0.00416557196599
Frame:138 jitter:-0.00166766051274
Frame:139 jitter:-0.00255052451116
Frame:140 jitter:-0.00439946059209
Frame:141 jitter:-0.00149671439153
Frame:142 jitter:-0.00454632643681
Frame:143 jitter:-0.00108758810979
Frame:144 jitter:-0.00220052603703
Frame:145 jitter:-0.00374762419682
Frame:146 jitter:-0.00280753973943
Frame:147 jitter:-0.00263659361821
Frame:148 jitter:-0.00531045798283
Frame:149 jitter:-0.000538748540696
Frame:150 jitter:-0.00233547095281
Frame:151 jitter:-0.00357047919255
Frame:152 jitter:0.00120933648128
Frame:153 jitter:-0.00279847983342
Frame:154 jitter:-0.00275055769902

Frame:155 jitter:-0.00424353484136
Frame:156 jitter:-0.00175659064275
average framerate = 0.162860554495
worst case = 0.583452939987

Sobel Frame:
Frame:199 jitter:-0.00407525857984
Frame:200 jitter:0.00422957578601
Frame:201 jitter:-0.00255057176648
Frame:202 jitter:-0.00314351877271
Frame:203 jitter:-0.000957458820926
Frame:204 jitter:-0.00213929971753
Frame:205 jitter:-0.00479266008435
Frame:206 jitter:0.000810653361691
Frame:207 jitter:-0.00198933443128
Frame:208 jitter:-0.00156566461621
Frame:209 jitter:-0.00248929819165
Frame:210 jitter:-0.00188252290784
Frame:211 jitter:-0.00220534166394
Frame:212 jitter:-0.00187346300183
Frame:213 jitter:-0.00212356409131
Frame:214 jitter:-0.00166651567517
Frame:215 jitter:-0.00235339960157
Frame:216 jitter:-0.00194951852857
Frame:217 jitter:-0.00213333925305
Frame:218 jitter:-0.00199553331433
Frame:219 jitter:-0.00208446344434
Frame:220 jitter:-0.00124427637159
Frame:221 jitter:-0.00262162050305
Frame:222 jitter:-0.00210234483777
Frame:223 jitter:-0.00161644777356
Frame:224 jitter:-0.00239559969007
Frame:225 jitter:0.000720531138791
Frame:226 jitter:-0.00230929216443
Frame:227 jitter:-0.000408619251834
Frame:228 jitter:-0.00202724298535
average framerate = 0.162014454213
worst case = 0.583709001541