University of Colorado
Boulder

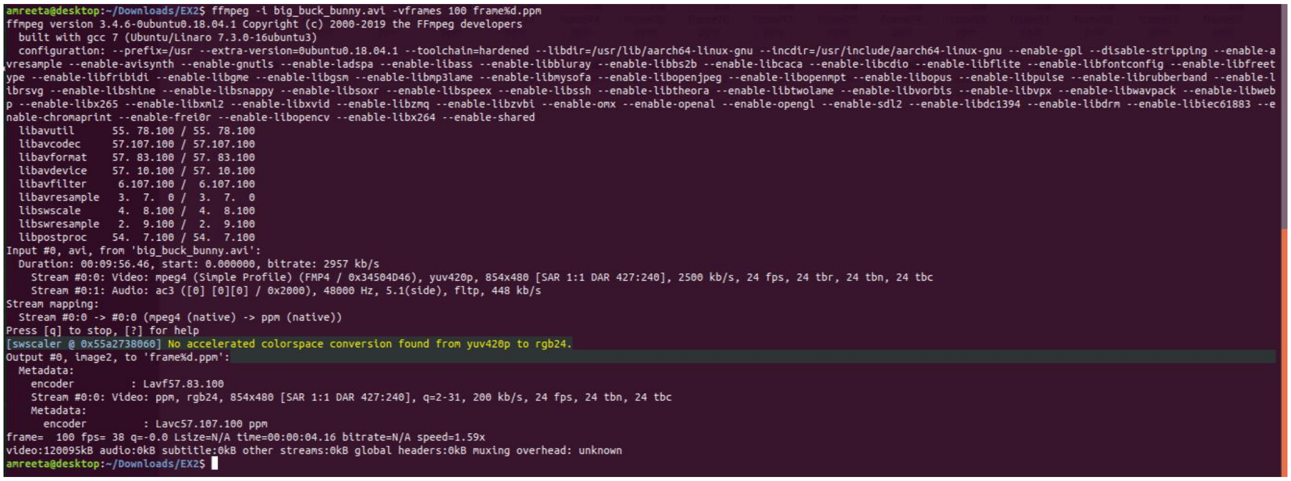
EXERCISE 2

BY
Amreeta Sengupta

06/21/2019
Question 1,2,3,4 have been performed using Jetson Nano

QUESTION 1

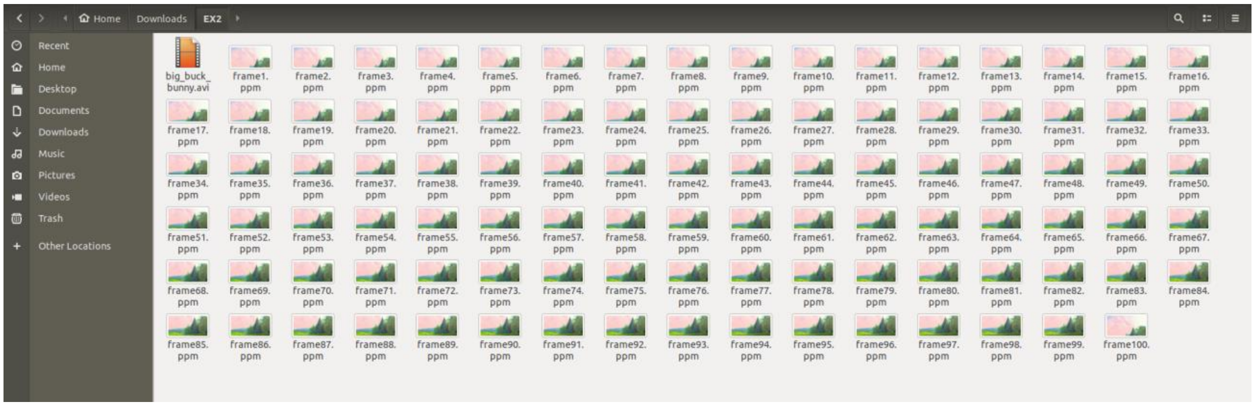
Ffmpeg command to extract frames



```
amreeta@desktop:~/Downloads/EX2$ ffmpeg -i big_buck_bunny.avi -vframes 100 frame%d.ppm
ffmpeg version 3.4.6-0ubuntu0.18.04.1 Copyright (c) 2000-2019 the FFmpeg developers
  built with gcc 7 (Ubuntu/Linaro 7.3.0-16ubuntu3)
  configuration: --prefix=/usr --extra-version=0ubuntu0.18.04.1 --toolchain=hardened --libdir=/usr/lib/aarch64-linux-gnu --incdir=/usr/include/aarch64-linux-gnu --enable-gpl --disable-stripping --enable-a
vresample --enable-avisynth --enable-gnutls --enable-ladspa --enable-libass --enable-libbluray --enable-libbs2b --enable-libcaca --enable-libcdio --enable-libflite --enable-libfontconfig --enable-libfreet
ype --enable-libfribidi --enable-libgme --enable-libgsm --enable-libmp3lame --enable-libmysofa --enable-libopenjpeg --enable-libopenmpt --enable-libopus --enable-libpulse --enable-librubberband --enable-l
ibrsvg --enable-libshine --enable-libsnappy --enable-libsoxr --enable-libspeex --enable-libssh --enable-libtheora --enable-libtwolame --enable-libvorbis --enable-libvpx --enable-libwavpack --enable-libweb
p --enable-libx265 --enable-libxml2 --enable-libxvid --enable-libzmq --enable-libzvbi --enable-omx --enable-openal --enable-opengl --enable-sdl2 --enable-libdc1394 --enable-libdrm --enable-libiec61883 --e
nable-chromaprint --enable-frei0r --enable-libopencv --enable-libx264 --enable-shared
  libavutil      55. 78.100 / 55. 78.100
  libavcodec     57.107.100 / 57.107.100
  libavformat    57. 83.100 / 57. 83.100
  libavdevice    57. 10.100 / 57. 10.100
  libavfilter     6.107.100 /  6.107.100
  libavresample   3.  7.  0 /  3.  7.  0
  libswscale      4.  8.100 /  4.  8.100
  libswresample   2.  9.100 /  2.  9.100
  libpostproc    54.  7.100 / 54.  7.100
Input #0, avi, from 'big_buck_bunny.avi':
  Duration: 00:09:56.46, start: 0.000000, bitrate: 2957 kb/s
    Stream #0:0: Video: mpeg4 (Simple Profile) (FMP4 / 0x34504D46), yuv420p, 854x480 [SAR 1:1 DAR 427:240], 2500 kb/s, 24 fps, 24 tbr, 24 tbn, 24 tbc
    Stream #0:1: Audio: ac3 ([0] [0][0] / 0x2000), 48000 Hz, 5.1(side), fltp, 448 kb/s
Stream mapping:
  Stream #0:0 -> #0:0 (mpeg4 (native) -> ppm (native))
Press [q] to stop, [?] for help
[swscaler @ 0x55a2738060] No accelerated colorspace conversion found from yuv420p to rgb24.
Output #0, image2, to 'frame%d.ppm':
  Metadata:
    encoder         : Lavf57.83.100
    Stream #0:0: Video: ppm, rgb24, 854x480 [SAR 1:1 DAR 427:240], q=2-31, 200 kb/s, 24 fps, 24 tbn, 24 tbc
    Metadata:
      encoder         : Lavc57.107.100 ppm
frame=  100 fps= 38 q=-0.0 Lsize=N/A time=00:00:04.16 bitrate=N/A speed=1.59x
video:120095kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: unknown
amreeta@desktop:~/Downloads/EX2$
```

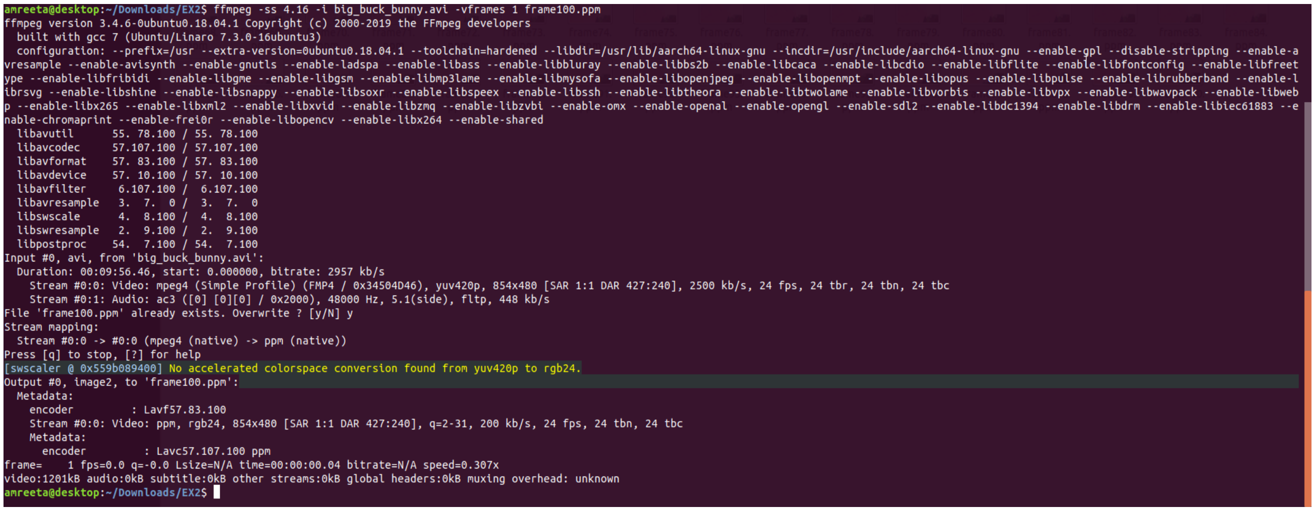
Here, -i specifies the input file which is big_buck_bunny.avi and -vframes is used to set the number of frames to be extracted and frames%d.ppm is the output file where %d will define each output file number.

Frames extracted using the command



Thus, 100 output files are extracted in the given location as shown above.

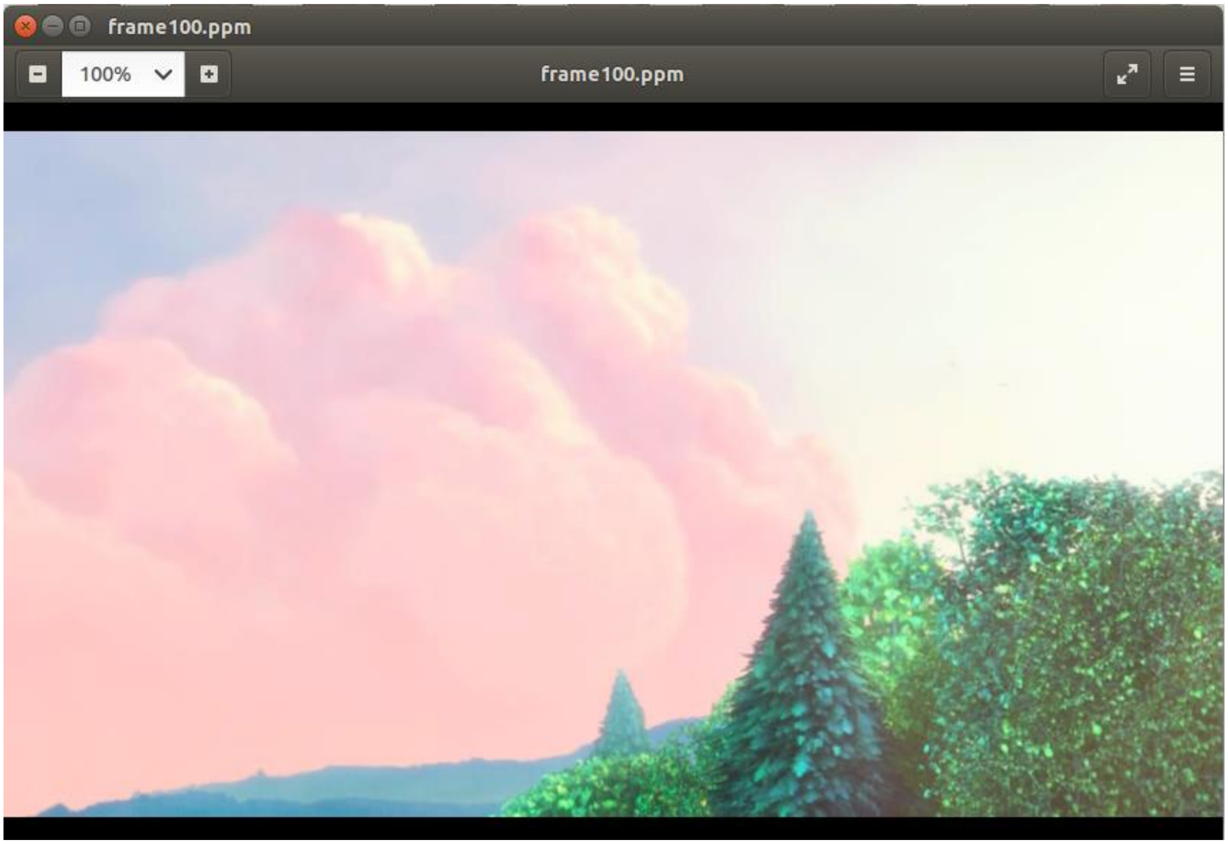
Ffmpeg command to extract 100th frame



```
amreeta@desktop:~/Downloads/EX2$ ffmpeg -ss 4.16 -i big_buck_bunny.avi -vframes 1 frame100.ppm
ffmpeg version 3.4.6-0ubuntu0.18.04.1 Copyright (c) 2000-2019 the FFmpeg developers
  built with gcc 7 (Ubuntu/Linaro 7.3.0-16ubuntu3)
  configuration: --prefix=/usr --extra-version=0ubuntu0.18.04.1 --toolchain=hardened --libdir=/usr/lib/aarch64-linux-gnu --incdir=/usr/include/aarch64-linux-gnu --enable-gpl --disable-stripping --enable-a
vresample --enable-avisynth --enable-gnutls --enable-ladspa --enable-libass --enable-libbluray --enable-libbs2b --enable-libcaca --enable-libcdio --enable-libflite --enable-libfontconfig --enable-libfreet
ype --enable-libfribidi --enable-libgme --enable-libgsm --enable-libmp3lame --enable-libmysofa --enable-libopenjpeg --enable-libopenmpt --enable-libopus --enable-libpulse --enable-librubberband --enable-l
ibrsvg --enable-libshine --enable-libsnappy --enable-libsoxr --enable-libspeex --enable-libssh --enable-libtheora --enable-libtwolame --enable-libvorbis --enable-libvpx --enable-libwavpack --enable-libweb
p --enable-libx265 --enable-libxml2 --enable-libxvid --enable-libzmq --enable-libzvbi --enable-omx --enable-openal --enable-opengl --enable-sdl2 --enable-libdc1394 --enable-libdrm --enable-libiec61883 --e
nable-chromaprint --enable-frei0r --enable-libopencv --enable-libx264 --enable-shared
  libavutil      55. 78.100 / 55. 78.100
  libavcodec     57.107.100 / 57.107.100
  libavformat    57. 83.100 / 57. 83.100
  libavdevice    57. 10.100 / 57. 10.100
  libavfilter     6.107.100 /  6.107.100
  libavresample   3.  7.  0 /  3.  7.  0
  libswscale      4.  8.100 /  4.  8.100
  libswresample   2.  9.100 /  2.  9.100
  libpostproc    54.  7.100 / 54.  7.100
Input #0, avi, from 'big_buck_bunny.avi':
  Duration: 00:09:56.46, start: 0.000000, bitrate: 2957 kb/s
    Stream #0:0: Video: mpeg4 (Simple Profile) (FMP4 / 0x34504D46), yuv420p, 854x480 [SAR 1:1 DAR 427:240], 2500 kb/s, 24 fps, 24 tbr, 24 tbn, 24 tbc
    Stream #0:1: Audio: ac3 ([0] [0][0] / 0x2000), 48000 Hz, 5.1(side), fltp, 448 kb/s
File 'frame100.ppm' already exists. Overwrite ? [y/N] y
Stream mapping:
  Stream #0:0 -> #0:0 (mpeg4 (native) -> ppm (native))
Press [q] to stop, [?] for help
[swscaler @ 0x559b089400] No accelerated colorspace conversion found from yuv420p to rgb24.
Output #0, image2, to 'frame100.ppm':
  Metadata:
    encoder         : Lavf57.83.100
    Stream #0:0: Video: ppm, rgb24, 854x480 [SAR 1:1 DAR 427:240], q=2-31, 200 kb/s, 24 fps, 24 tbn, 24 tbc
    Metadata:
      encoder         : Lavc57.107.100 ppm
frame=    1 fps=0.0 q=-0.0 Lsize=N/A time=00:00:00.04 bitrate=N/A speed=0.307x
video:1201kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: unknown
amreeta@desktop:~/Downloads/EX2$
```

Here, -ss specifies the time from which the frame should be extracted. Since FPS is 24Hz, therefore, to extract the 100th frame, we can calculate it to be 100/24 = 4.16 seconds and hence we use 4.16 as the start time. -i specifies the input file which is big_buck_bunny.avi and -vframes is used to set the number of frames to be extracted which will be 1 in this case and frames100.ppm is the output file.

100th Frame extracted



QUESTION 2

Original Image

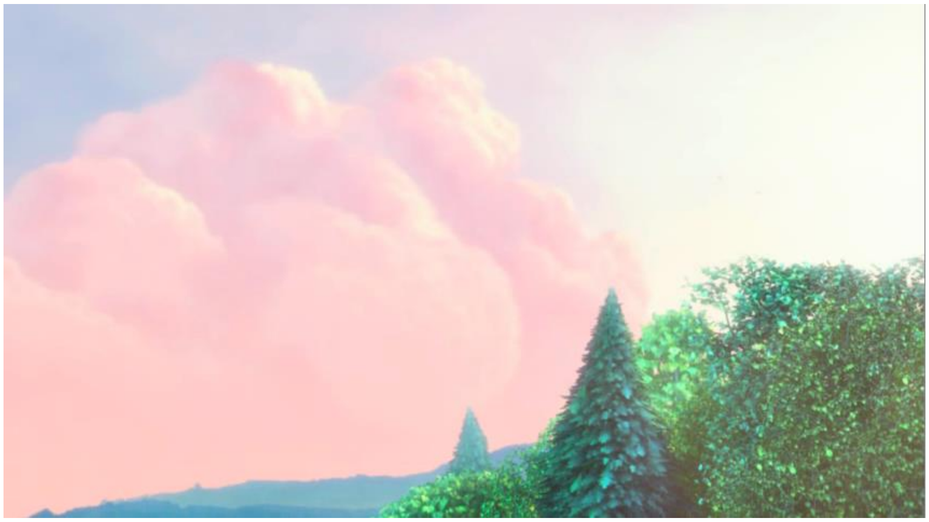
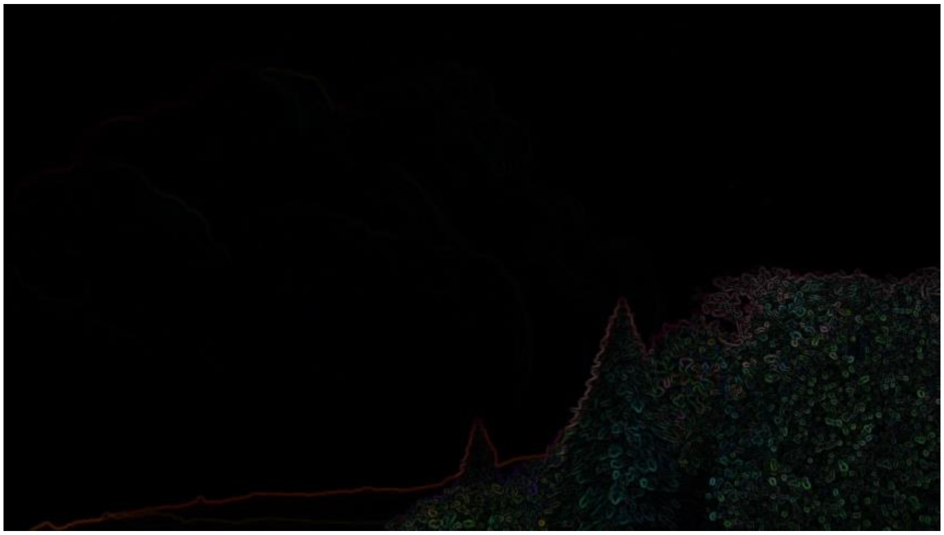


Image after application of Sobel Transform



Options used to create the transform

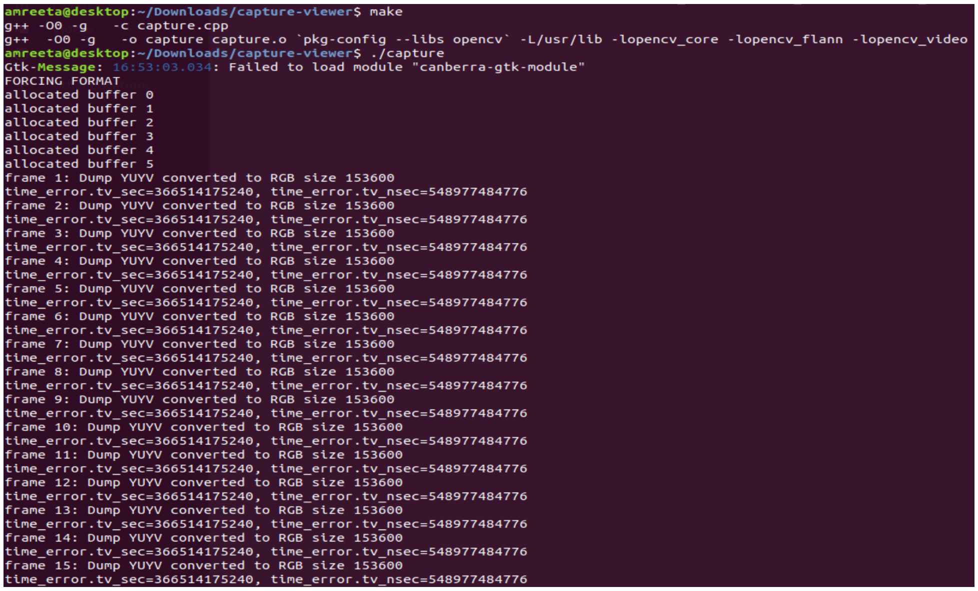
- Open GIMP and open the image on which the transform must be performed.
- From the option bar on the top, select Filters to apply the Sobel Transform.
- Then, from the list of options, select Edge-Detect as Sobel Transform is an edge enhancement transform.
- Next, click on Sobel from the menu of options.
- Now check the following boxes in order to apply the Sobel Transform in both X and Y direction:
  - Preview

  - Sobel horizontally
  - Sobel vertically
- After these settings, click on OK.
- Thus, the Sobel Transform will be applied on the image.

<u>QUESTION 3</u>

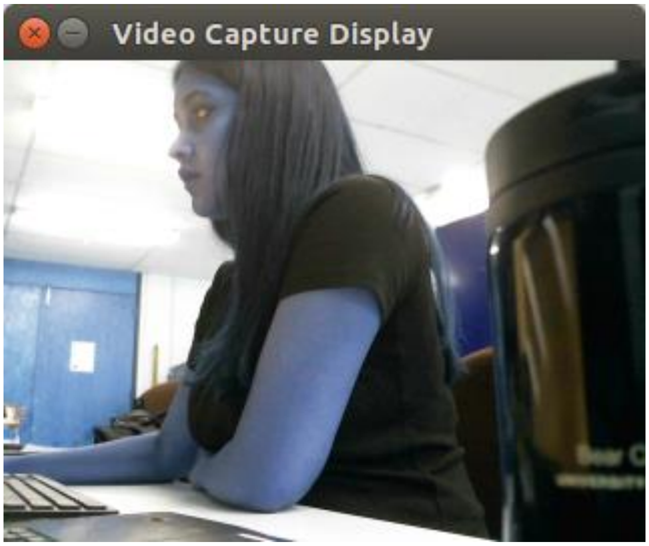
<u>Capture-viewer</u>

- <u>Build and Run</u>



```
amreeta@desktop:~/Downloads/capture-viewer$ make
g++ -O0 -g   -c capture.cpp
g++  -O0 -g   -o capture capture.o `pkg-config --libs opencv` -L/usr/lib -lopencv_core -lopencv_flann -lopencv_video
amreeta@desktop:~/Downloads/capture-viewer$ ./capture
Gtk-Message: 16:53:03.034: Failed to load module "canberra-gtk-module"
FORCING FORMAT
allocated buffer 0
allocated buffer 1
allocated buffer 2
allocated buffer 3
allocated buffer 4
allocated buffer 5
frame 1: Dump YUYV converted to RGB size 153600
time_error.tv_sec=366514175240, time_error.tv_nsec=548977484776
frame 2: Dump YUYV converted to RGB size 153600
time_error.tv_sec=366514175240, time_error.tv_nsec=548977484776
frame 3: Dump YUYV converted to RGB size 153600
time_error.tv_sec=366514175240, time_error.tv_nsec=548977484776
frame 4: Dump YUYV converted to RGB size 153600
time_error.tv_sec=366514175240, time_error.tv_nsec=548977484776
frame 5: Dump YUYV converted to RGB size 153600
time_error.tv_sec=366514175240, time_error.tv_nsec=548977484776
frame 6: Dump YUYV converted to RGB size 153600
time_error.tv_sec=366514175240, time_error.tv_nsec=548977484776
frame 7: Dump YUYV converted to RGB size 153600
time_error.tv_sec=366514175240, time_error.tv_nsec=548977484776
frame 8: Dump YUYV converted to RGB size 153600
time_error.tv_sec=366514175240, time_error.tv_nsec=548977484776
frame 9: Dump YUYV converted to RGB size 153600
time_error.tv_sec=366514175240, time_error.tv_nsec=548977484776
frame 10: Dump YUYV converted to RGB size 153600
time_error.tv_sec=366514175240, time_error.tv_nsec=548977484776
frame 11: Dump YUYV converted to RGB size 153600
time_error.tv_sec=366514175240, time_error.tv_nsec=548977484776
frame 12: Dump YUYV converted to RGB size 153600
time_error.tv_sec=366514175240, time_error.tv_nsec=548977484776
frame 13: Dump YUYV converted to RGB size 153600
time_error.tv_sec=366514175240, time_error.tv_nsec=548977484776
frame 14: Dump YUYV converted to RGB size 153600
time_error.tv_sec=366514175240, time_error.tv_nsec=548977484776
frame 15: Dump YUYV converted to RGB size 153600
time_error.tv_sec=366514175240, time_error.tv_nsec=548977484776
```

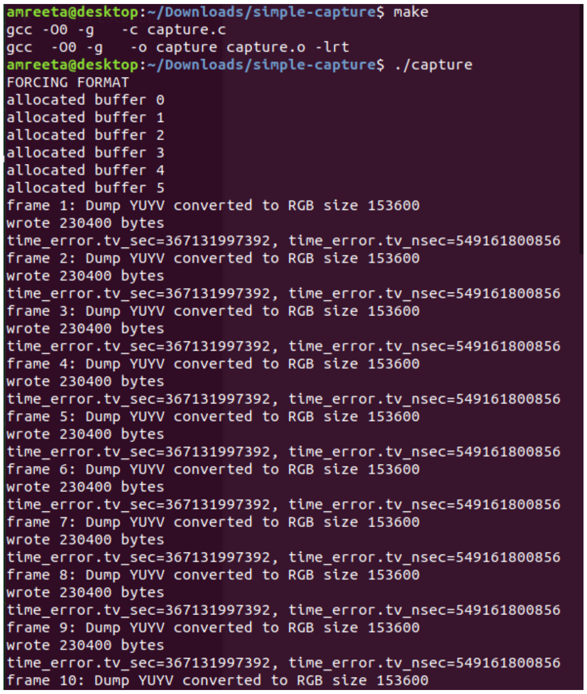
- <u>Code Output</u>



This code expects runtime argument from user either in short or long form. These arguments are used to set device name, print help menu, use memory mapped buffers [default], use read system call to access the camera device driver, extract number of frames specified by the user, forcefully format the image to 640X480 resolution in greyscale. This code uses open(), ioctl(), close() system call to access device drivers of the camera to capture image. This basically replaces the OpenCV API to capture the image. Based on runtime switch, ioctl sys call is used to verify whether those capabilities are provided by the driver or not. In case of successful verification, it processes the image and displays it. This code continuously captures, processes and displays the captured image.

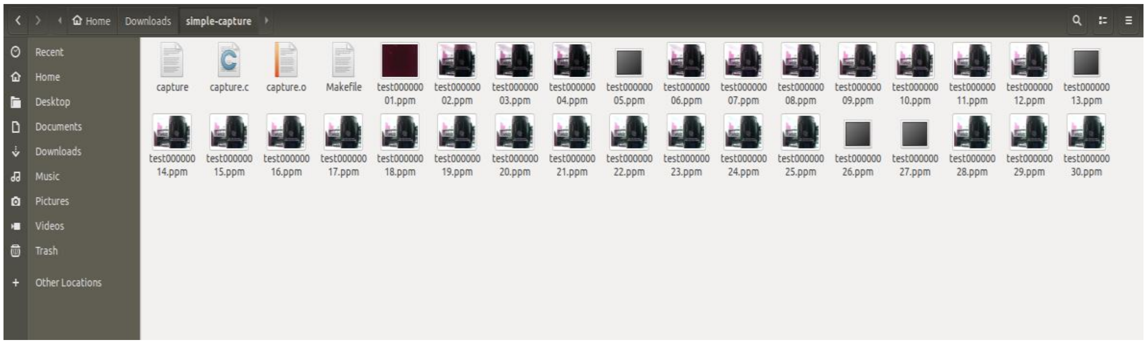
Simple-capture

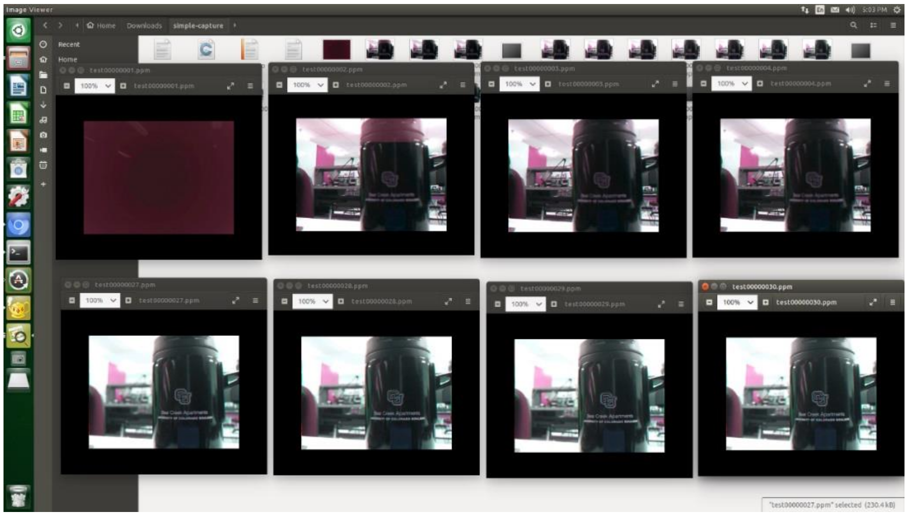
- Build and Run



```
amreeta@desktop:~/Downloads/simple-capture$ make
gcc -O0 -g   -c capture.c
gcc  -O0 -g   -o capture capture.o -lrt
amreeta@desktop:~/Downloads/simple-capture$ ./capture
FORCING FORMAT
allocated buffer 0
allocated buffer 1
allocated buffer 2
allocated buffer 3
allocated buffer 4
allocated buffer 5
frame 1: Dump YUYV converted to RGB size 153600
wrote 230400 bytes
time_error.tv_sec=367131997392, time_error.tv_nsec=549161800856
frame 2: Dump YUYV converted to RGB size 153600
wrote 230400 bytes
time_error.tv_sec=367131997392, time_error.tv_nsec=549161800856
frame 3: Dump YUYV converted to RGB size 153600
wrote 230400 bytes
time_error.tv_sec=367131997392, time_error.tv_nsec=549161800856
frame 4: Dump YUYV converted to RGB size 153600
wrote 230400 bytes
time_error.tv_sec=367131997392, time_error.tv_nsec=549161800856
frame 5: Dump YUYV converted to RGB size 153600
wrote 230400 bytes
time_error.tv_sec=367131997392, time_error.tv_nsec=549161800856
frame 6: Dump YUYV converted to RGB size 153600
wrote 230400 bytes
time_error.tv_sec=367131997392, time_error.tv_nsec=549161800856
frame 7: Dump YUYV converted to RGB size 153600
wrote 230400 bytes
time_error.tv_sec=367131997392, time_error.tv_nsec=549161800856
frame 8: Dump YUYV converted to RGB size 153600
wrote 230400 bytes
time_error.tv_sec=367131997392, time_error.tv_nsec=549161800856
frame 9: Dump YUYV converted to RGB size 153600
wrote 230400 bytes
time_error.tv_sec=367131997392, time_error.tv_nsec=549161800856
frame 10: Dump YUYV converted to RGB size 153600
```

- Code Output

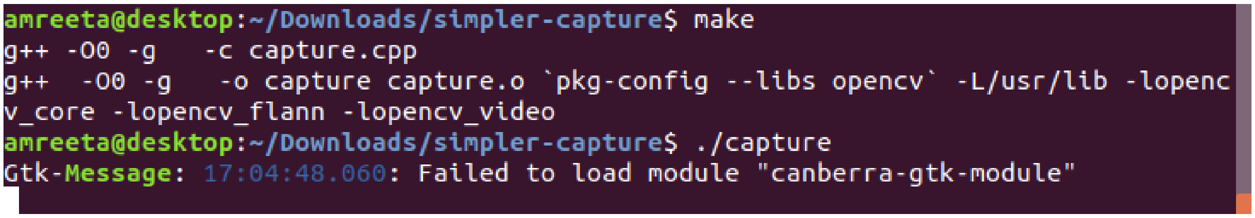




This code is similar to the previous one and it also expects runtime argument in order to format the image in various ways including setting the device name, printing the help menu, extracting the number of frames specified by the user etc. It employs open(), ioctl(), close() system calls to access the camera's device drivers for capturing the image. On successful verification through use of ioctl sys, it processes the image and displays it.

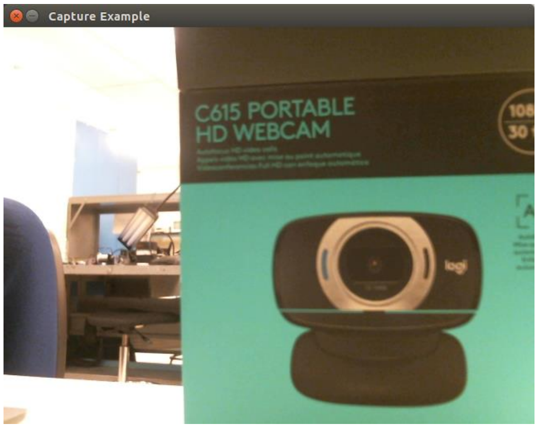
Simpler-capture

- Build and Run



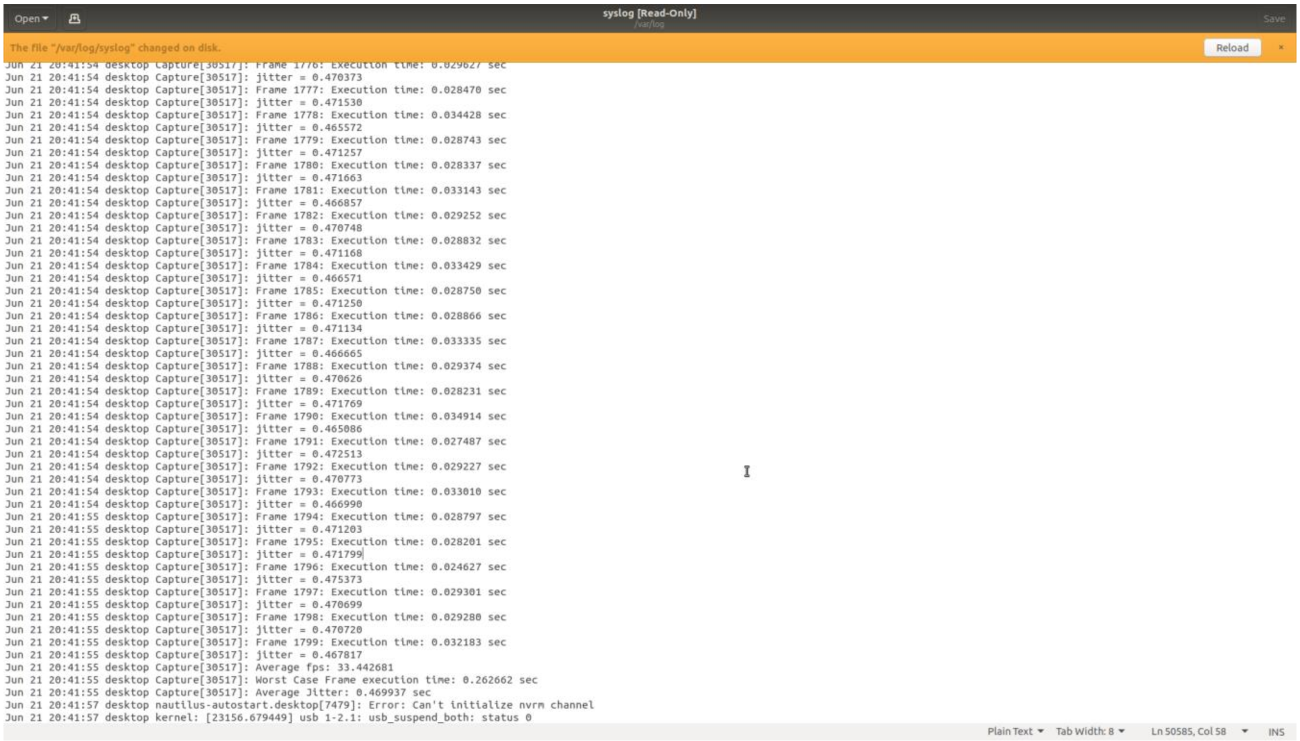
```
amreeta@desktop:~/Downloads/simpler-capture$ make
g++ -O0 -g    -c capture.cpp
g++  -O0 -g    -o capture capture.o `pkg-config --libs opencv` -L/usr/lib -lopenc
v_core -lopencv_flann -lopencv_video
amreeta@desktop:~/Downloads/simpler-capture$ ./capture
Gtk-Message: 17:04:48.060: Failed to load module "canberra-gtk-module"
```

- Code Output



This code is used for continuous streaming of image frames using an infinite loop in which the image frame is captured, and it is checked if it is NULL which would result in break. cvNamedWindow API is used to create a window called Capture Example and cvShowImage API is used to display the image in this named window. ESC key can be used to terminate the program.

Logs made in the modified code



The file "/var/log/syslog" changed on disk.

```
Jun 21 20:41:54 desktop Capture[30517]: Frame 1776: Execution time: 0.029627 sec
Jun 21 20:41:54 desktop Capture[30517]: jitter = 0.470373
Jun 21 20:41:54 desktop Capture[30517]: Frame 1777: Execution time: 0.028470 sec
Jun 21 20:41:54 desktop Capture[30517]: jitter = 0.471530
Jun 21 20:41:54 desktop Capture[30517]: Frame 1778: Execution time: 0.034428 sec
Jun 21 20:41:54 desktop Capture[30517]: jitter = 0.465572
Jun 21 20:41:54 desktop Capture[30517]: Frame 1779: Execution time: 0.028743 sec
Jun 21 20:41:54 desktop Capture[30517]: jitter = 0.471257
Jun 21 20:41:54 desktop Capture[30517]: Frame 1780: Execution time: 0.028337 sec
Jun 21 20:41:54 desktop Capture[30517]: jitter = 0.471663
Jun 21 20:41:54 desktop Capture[30517]: Frame 1781: Execution time: 0.033143 sec
Jun 21 20:41:54 desktop Capture[30517]: jitter = 0.466857
Jun 21 20:41:54 desktop Capture[30517]: Frame 1782: Execution time: 0.029252 sec
Jun 21 20:41:54 desktop Capture[30517]: jitter = 0.470748
Jun 21 20:41:54 desktop Capture[30517]: Frame 1783: Execution time: 0.028832 sec
Jun 21 20:41:54 desktop Capture[30517]: jitter = 0.471168
Jun 21 20:41:54 desktop Capture[30517]: Frame 1784: Execution time: 0.033429 sec
Jun 21 20:41:54 desktop Capture[30517]: jitter = 0.466571
Jun 21 20:41:54 desktop Capture[30517]: Frame 1785: Execution time: 0.028750 sec
Jun 21 20:41:54 desktop Capture[30517]: jitter = 0.471250
Jun 21 20:41:54 desktop Capture[30517]: Frame 1786: Execution time: 0.028866 sec
Jun 21 20:41:54 desktop Capture[30517]: jitter = 0.471134
Jun 21 20:41:54 desktop Capture[30517]: Frame 1787: Execution time: 0.033335 sec
Jun 21 20:41:54 desktop Capture[30517]: jitter = 0.466665
Jun 21 20:41:54 desktop Capture[30517]: Frame 1788: Execution time: 0.029374 sec
Jun 21 20:41:54 desktop Capture[30517]: jitter = 0.470626
Jun 21 20:41:54 desktop Capture[30517]: Frame 1789: Execution time: 0.028231 sec
Jun 21 20:41:54 desktop Capture[30517]: jitter = 0.471769
Jun 21 20:41:54 desktop Capture[30517]: Frame 1790: Execution time: 0.034914 sec
Jun 21 20:41:54 desktop Capture[30517]: jitter = 0.465086
Jun 21 20:41:54 desktop Capture[30517]: Frame 1791: Execution time: 0.027487 sec
Jun 21 20:41:54 desktop Capture[30517]: jitter = 0.472513
Jun 21 20:41:54 desktop Capture[30517]: Frame 1792: Execution time: 0.029227 sec
Jun 21 20:41:54 desktop Capture[30517]: jitter = 0.470773
Jun 21 20:41:54 desktop Capture[30517]: Frame 1793: Execution time: 0.033010 sec
Jun 21 20:41:54 desktop Capture[30517]: jitter = 0.466990
Jun 21 20:41:55 desktop Capture[30517]: Frame 1794: Execution time: 0.028797 sec
Jun 21 20:41:55 desktop Capture[30517]: jitter = 0.471203
Jun 21 20:41:55 desktop Capture[30517]: Frame 1795: Execution time: 0.028201 sec
Jun 21 20:41:55 desktop Capture[30517]: jitter = 0.471799
Jun 21 20:41:55 desktop Capture[30517]: Frame 1796: Execution time: 0.024627 sec
Jun 21 20:41:55 desktop Capture[30517]: jitter = 0.475373
Jun 21 20:41:55 desktop Capture[30517]: Frame 1797: Execution time: 0.029301 sec
Jun 21 20:41:55 desktop Capture[30517]: jitter = 0.470699
Jun 21 20:41:55 desktop Capture[30517]: Frame 1798: Execution time: 0.029280 sec
Jun 21 20:41:55 desktop Capture[30517]: jitter = 0.470720
Jun 21 20:41:55 desktop Capture[30517]: Frame 1799: Execution time: 0.032183 sec
Jun 21 20:41:55 desktop Capture[30517]: jitter = 0.467817
Jun 21 20:41:55 desktop Capture[30517]: Average fps: 33.442681
Jun 21 20:41:55 desktop Capture[30517]: Worst Case Frame execution time: 0.262662 sec
Jun 21 20:41:55 desktop Capture[30517]: Average Jitter: 0.469937 sec
Jun 21 20:41:57 desktop nautilus-autostart.desktop[7479]: Error: Can't initialize nvrm channel
Jun 21 20:41:57 desktop kernel: [23156.679449] usb 1-2.1: usb_suspend_both: status 0
```

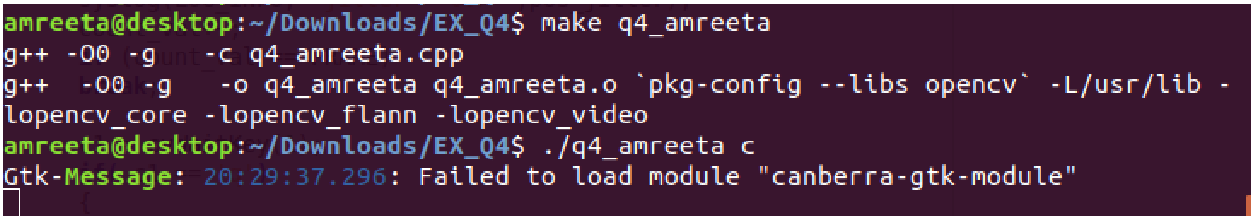
Frame Rate, Worst Case Execution Time, Jitter analysis

- Here, clock_gettime system call is used to get the time which gives a higher precision as compared to gettimeofday. CLOCK_REALTIME is used to return the number of seconds and nanoseconds.
- For Frame rate analysis, time taken between the start of the capture of frames by the camera and display of the image is utilized which gives us the total execution time for one frame. This is done for 1800 frames to determine the average frame rate.
- One divided by the execution time gives us the FPS and sum of these FPS values for each frame divided by the number of frames used gives us the average frame rate which is 33.4fps for this case.
- The worst-case execution time is determined by taking the maximum value of the calculated execution time and storing it in a variable.
- This will help us determine the deadline which is basically the worst-case execution time along with the addition of some safety margin.
- Here, since WCET = 0.262 seconds, hence, deadline is taken as 0.5 seconds

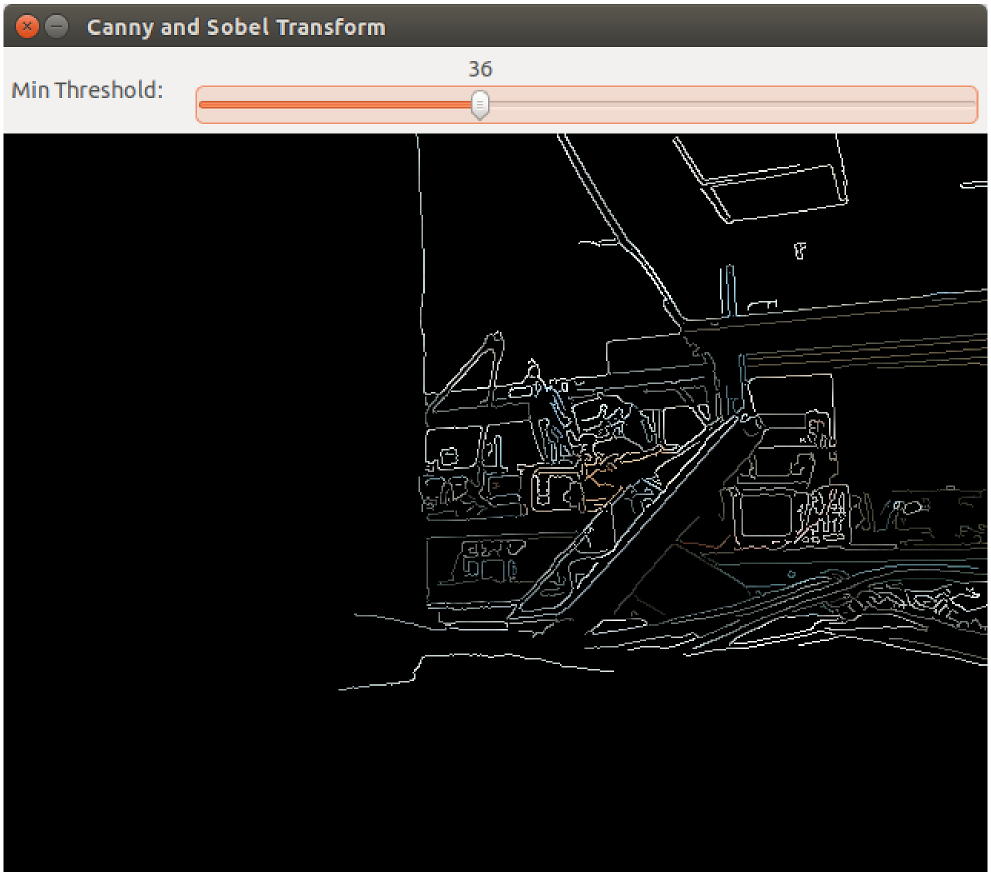
- The jitter is calculated as the difference between the calculated deadline and the execution time for each frame.
- The total jitter is basically the addition of the jitter calculated for each frame.
- Average jitter is given by the total jitter divided by the frame count which is 0.46 seconds for this case.
- All the calculated information is logged.
- Owing to a preemptible kernel in Linux, the value of the execution time may vary as the OS can schedule other tasks.

QUESTION 4

Canny Transform

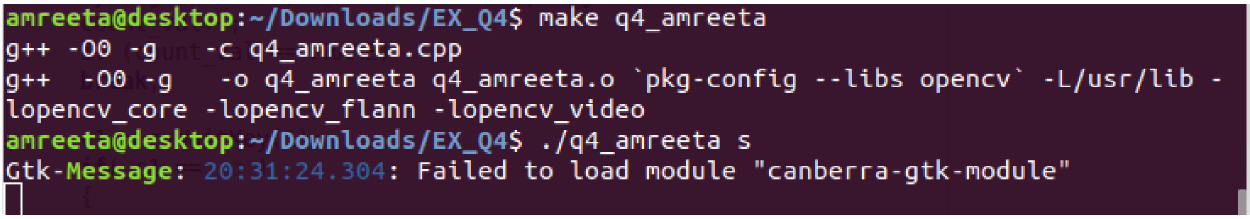


```
amreeta@desktop:~/Downloads/EX_Q4$ make q4_amreeta
g++ -O0 -g    -c q4_amreeta.cpp
g++  -O0 -g   -o q4_amreeta q4_amreeta.o `pkg-config --libs opencv` -L/usr/lib -lopencv_core -lopencv_flann -lopencv_video
amreeta@desktop:~/Downloads/EX_Q4$ ./q4_amreeta c
Gtk-Message: 20:29:37.296: Failed to load module "canberra-gtk-module"
```

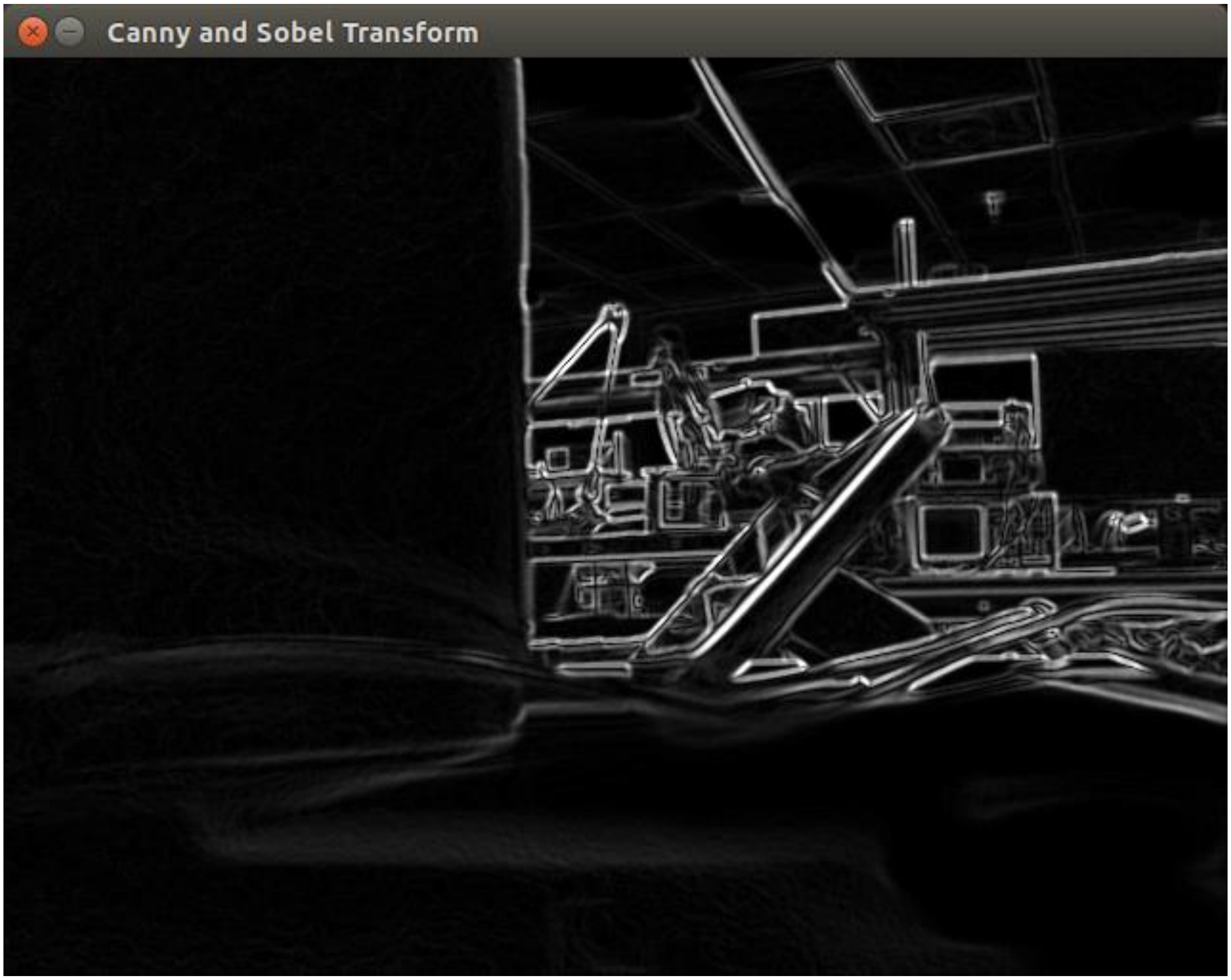


Min Threshold: 36

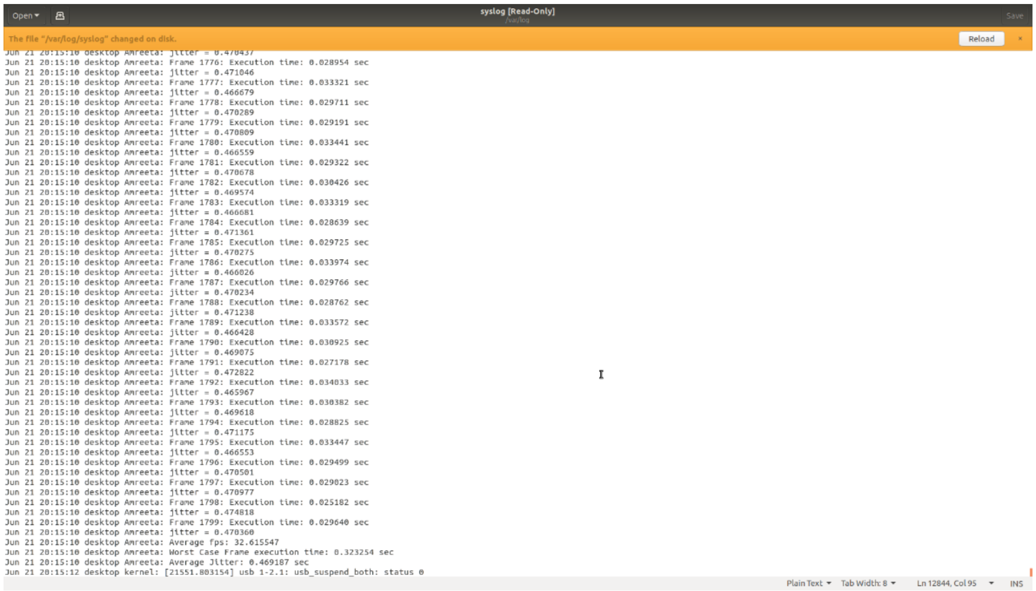
Sobel Transform



```
amreeta@desktop:~/Downloads/EX_Q4$ make q4_amreeta
g++ -O0 -g    -c q4_amreeta.cpp
g++  -O0 -g    -o q4_amreeta q4_amreeta.o `pkg-config --libs opencv` -L/usr/lib -
lopencv_core -lopencv_flann -lopencv_video
amreeta@desktop:~/Downloads/EX_Q4$ ./q4_amreeta s
Gtk-Message: 20:31:24.304: Failed to load module "canberra-gtk-module"
```



<u>Logs</u>



The file "/var/log/syslog" changed on disk.

```
Jun 21 20:15:10 desktop Amreeta: jitter = 0.470437
Jun 21 20:15:10 desktop Amreeta: Frame 1776: Execution time: 0.028954 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.471046
Jun 21 20:15:10 desktop Amreeta: Frame 1777: Execution time: 0.033321 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.466679
Jun 21 20:15:10 desktop Amreeta: Frame 1778: Execution time: 0.029711 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.470289
Jun 21 20:15:10 desktop Amreeta: Frame 1779: Execution time: 0.029191 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.470809
Jun 21 20:15:10 desktop Amreeta: Frame 1780: Execution time: 0.033441 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.466559
Jun 21 20:15:10 desktop Amreeta: Frame 1781: Execution time: 0.029322 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.470678
Jun 21 20:15:10 desktop Amreeta: Frame 1782: Execution time: 0.030426 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.469574
Jun 21 20:15:10 desktop Amreeta: Frame 1783: Execution time: 0.033319 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.466681
Jun 21 20:15:10 desktop Amreeta: Frame 1784: Execution time: 0.028639 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.471361
Jun 21 20:15:10 desktop Amreeta: Frame 1785: Execution time: 0.029725 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.470275
Jun 21 20:15:10 desktop Amreeta: Frame 1786: Execution time: 0.033974 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.466026
Jun 21 20:15:10 desktop Amreeta: Frame 1787: Execution time: 0.029766 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.470234
Jun 21 20:15:10 desktop Amreeta: Frame 1788: Execution time: 0.028762 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.471238
Jun 21 20:15:10 desktop Amreeta: Frame 1789: Execution time: 0.033572 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.466428
Jun 21 20:15:10 desktop Amreeta: Frame 1790: Execution time: 0.030925 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.469075
Jun 21 20:15:10 desktop Amreeta: Frame 1791: Execution time: 0.027178 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.472822
Jun 21 20:15:10 desktop Amreeta: Frame 1792: Execution time: 0.034033 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.465967
Jun 21 20:15:10 desktop Amreeta: Frame 1793: Execution time: 0.030382 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.469618
Jun 21 20:15:10 desktop Amreeta: Frame 1794: Execution time: 0.028825 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.471175
Jun 21 20:15:10 desktop Amreeta: Frame 1795: Execution time: 0.033447 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.466553
Jun 21 20:15:10 desktop Amreeta: Frame 1796: Execution time: 0.029499 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.470501
Jun 21 20:15:10 desktop Amreeta: Frame 1797: Execution time: 0.029023 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.470977
Jun 21 20:15:10 desktop Amreeta: Frame 1798: Execution time: 0.025182 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.474818
Jun 21 20:15:10 desktop Amreeta: Frame 1799: Execution time: 0.029640 sec
Jun 21 20:15:10 desktop Amreeta: jitter = 0.470360
Jun 21 20:15:10 desktop Amreeta: Average fps: 32.615547
Jun 21 20:15:10 desktop Amreeta: Worst Case Frame execution time: 0.323254 sec
Jun 21 20:15:10 desktop Amreeta: Average Jitter: 0.469187 sec
Jun 21 20:15:12 desktop kernel: [21551.803154] usb 1-2.1: usb_suspend_both: status 0
```

Plain Text ▾ Tab Width: 8 ▾ Ln 12844, Col 95 ▾ INS

<u>Frame Rate, Worst Case Execution Time, Jitter analysis</u>

- In this program, the canny or sobel transform can be applied by the user which will be given as a run time argument i.e. 'c' or 'C' for canny transform and 's' or 'S' for sobel transform.
- Here, clock_gettime system call is used to get the time which gives a higher precision as compared to gettimeofday. CLOCK_REALTIME is used to return the number of seconds and nanoseconds.
- For Frame rate analysis, time taken between the start of the capture of frames by the camera and display of the image is utilized which gives us the total execution time for one frame. This is done for 1800 frames to determine the average frame rate.
- One divided by the execution time gives us the FPS and sum of these FPS values for each frame divided by the number of frames used gives us the average frame rate which is 32.61fps for this case.
- The worst-case execution time is determined by taking the maximum value of the calculated execution time and storing it in a variable.
- This will help us determine the deadline which is basically the worst-case execution time along with the addition of some safety margin.
- Here, since WCET = 0.262 seconds, hence, deadline is taken as 0.5 seconds
- The jitter is calculated as the difference between the calculated deadline and the execution time for each frame.
- The total jitter is basically the addition of the jitter calculated for each frame.
- Average jitter is given by the total jitter divided by the frame count which is 0.46 seconds for thos case.
- All the calculated information is logged.
- Owing to a preemptible kernel in Linux, the value of the execution time may vary as the OS can schedule other tasks.

REFERENCES

- http://www.cse.uaa.alaska.edu/~ssiewert/a485_code/
- https://docs.opencv.org
- RTES Exercise 4 by Amreeta Sengupta