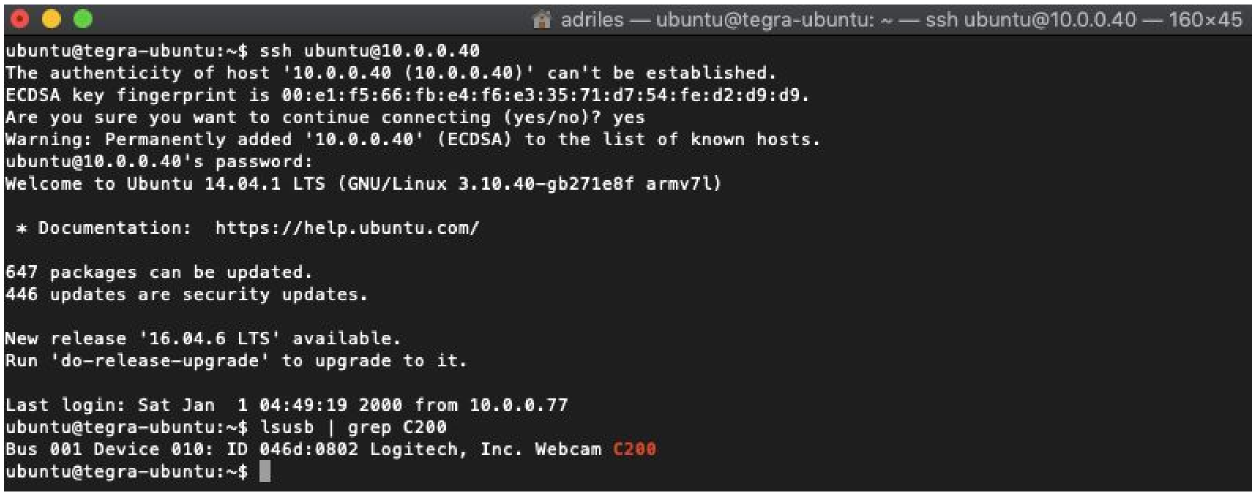
Adrian Unkeles
Sarthak Jain
ECEN 5623
Exercise 4
Board: Jetson TK1

Part 1

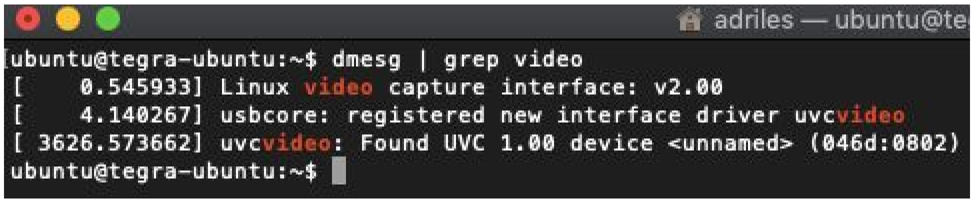
Here we simply needed to verify that the Logitech Webcam C200 is able to connect and be recognized by the Jetson TK1. We use the “lsusb” kernel driver configuration tool as seen below to verify that the camera is recognized:


adriles — ubuntu@tegra-ubuntu: ~ — ssh ubuntu@10.0.0.40 — 160×45

```
ubuntu@tegra-ubuntu:~$ ssh ubuntu@10.0.0.40
The authenticity of host '10.0.0.40 (10.0.0.40)' can't be established.
ECDSA key fingerprint is 00:e1:f5:66:fb:e4:f6:e3:35:71:d7:54:fe:d2:d9:d9.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.0.40' (ECDSA) to the list of known hosts.
ubuntu@10.0.0.40's password:
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.10.40-gb271e8f armv7l)

 * Documentation:  https://help.ubuntu.com/

647 packages can be updated.
446 updates are security updates.

New release '16.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sat Jan  1 04:49:19 2000 from 10.0.0.77
ubuntu@tegra-ubuntu:~$ lsusb | grep C200
Bus 001 Device 010: ID 046d:0802 Logitech, Inc. Webcam C200
ubuntu@tegra-ubuntu:~$
```

Figure 1: Showing that the Webcam C200 device is recognized as a connected device to the USB bus.

We then also use the “dmesg” command to see that the UVC driver is loaded:



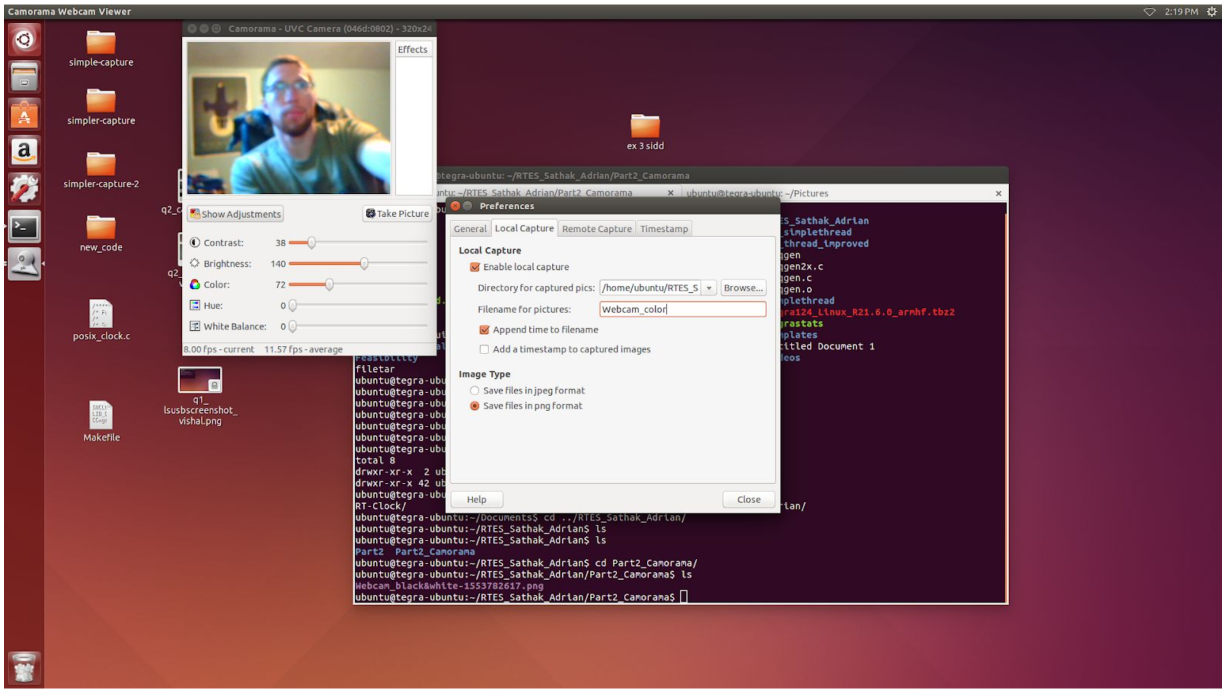
```
[ubuntu@tegra-ubuntu:~$ dmesg | grep video
[    0.545933] Linux video capture interface: v2.00
[    4.140267] usbcore: registered new interface driver uvcvideo
[ 3626.573662] uvcvideo: Found UVC 1.00 device <unnamed> (046d:0802)
ubuntu@tegra-ubuntu:~$
```

Figure 2: Showing that the kernel recognized a new interface driver “uvcvideo” and registered it with the usbcore. Furthermore, the “uvcvideo” driver found a UVC 1.00 device, which is not explicitly said to be our Webcam C200, but implicitly must be.

Part 2

Here we needed to install either Camorama or Cheese and do a simple snapshot capture. This proved to us that the camera had at least its basic functionality. Camorama and Cheese were already installed on this Jetson, but I did have to uninstall the gnome desktop and install the gtk desktop to make Camorama functional. I did not use Cheese once Camorama was working.

Screenshots of Camorama, as well as a few images in which I played with the color balance are shown below:



Contrast: 38
Brightness: 140
Color: 72
Hue: 0
White Balance: 0
8.00 fps - current   11.57 fps - average

Local Capture
[✓] Enable local capture
Directory for captured pics: /home/ubuntu/RTES_S
Filename for pictures: Webcam_color
[✓] Append time to filename
[ ] Add a timestamp to captured images

Image Type
Save files in jpeg format [✓ Save files in png format]

```
ubuntu@tegra-ubuntu:~/Documents$ cd ../RTES_Sathak_Adrian/
ubuntu@tegra-ubuntu:~/RTES_Sathak_Adrian$ ls
ubuntu@tegra-ubuntu:~/RTES_Sathak_Adrian$ ls
Part2  Part2_Camorama
ubuntu@tegra-ubuntu:~/RTES_Sathak_Adrian$ cd Part2_Camorama/
ubuntu@tegra-ubuntu:~/RTES_Sathak_Adrian/Part2_Camorama$ ls
Webcam_black&white-1553782617.png
ubuntu@tegra-ubuntu:~/RTES_Sathak_Adrian/Part2_Camorama$
```

Figure 3: Screenshot of Camorama installed and functional on the Jetson TK1

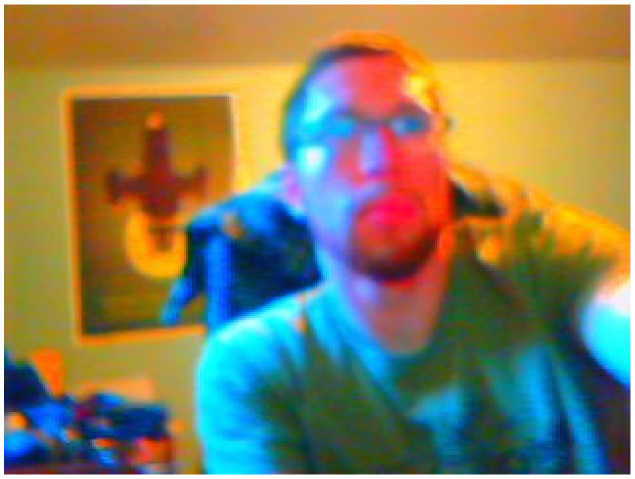


Figure 4: Snapshot with Camorama, turning up the color balance to maximum

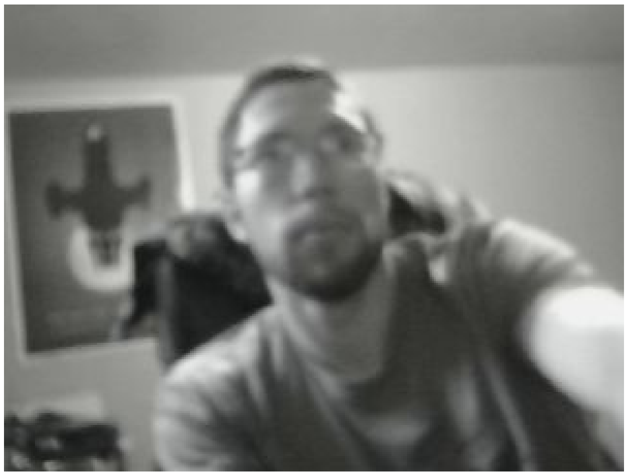


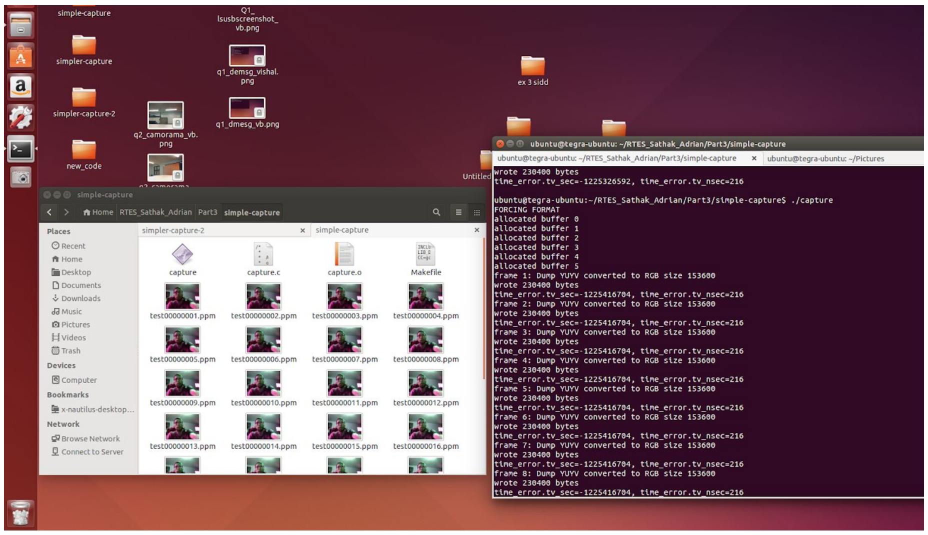
Figure 5: Snapshot with Camorama, turning down the color balance to minimum

Part 3

- Using your verified Logitech C200 camera on a DE1-SoC, Raspberry Pi or Jetson, verify that it can stream continuously using to a raw image buffer for transformation and processing using example code such as simple-capture, simpler-capture, or simpler-capture-2. Provide a screenshot to prove that you got continuous capture to work.

Compilation of simple-capture was straightforward since it doesn't require OpenCV. For screenshots of compiling simple-capture and simpler-capture, look in the Part3 folder of the Exercise 4 submission.

simple-capture



```
wrote 230400 bytes
time_error.tv_sec=-1225326592, time_error.tv_nsec=216

ubuntu@tegra-ubuntu:~/RTES_Sathak_Adrian/Part3/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=-1225416704, time_error.tv_nsec=216
frame 2: Dump YUYV converted to RGB size 153600
wrote 230400 bytes
time_error.tv_sec=-1225416704, time_error.tv_nsec=216
frame 3: Dump YUYV converted to RGB size 153600
wrote 230400 bytes
time_error.tv_sec=-1225416704, time_error.tv_nsec=216
frame 4: Dump YUYV converted to RGB size 153600
wrote 230400 bytes
time_error.tv_sec=-1225416704, time_error.tv_nsec=216
frame 5: Dump YUYV converted to RGB size 153600
wrote 230400 bytes
time_error.tv_sec=-1225416704, time_error.tv_nsec=216
frame 6: Dump YUYV converted to RGB size 153600
wrote 230400 bytes
time_error.tv_sec=-1225416704, time_error.tv_nsec=216
frame 7: Dump YUYV converted to RGB size 153600
wrote 230400 bytes
time_error.tv_sec=-1225416704, time_error.tv_nsec=216
frame 8: Dump YUYV converted to RGB size 153600
wrote 230400 bytes
time_error.tv_sec=-1225416704, time_error.tv_nsec=216
```

Figure 6: Running simple-capture, a stream of images written to the disk

The output stream of images from simple-capture are included in the submission folder at: Part3/streaming_test1:simple-capture. The output files from the streaming capture are .ppm files.

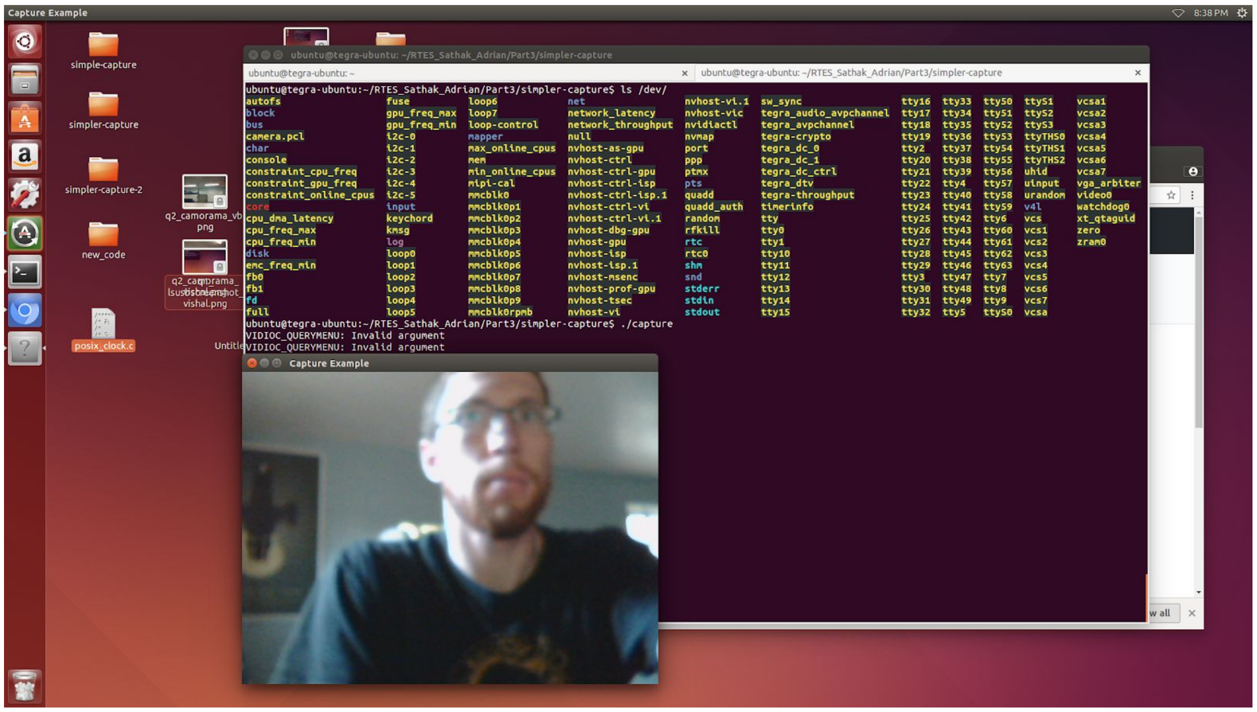
simpler-capture

Using simpler-capture required setup of OpenCV, which started a good deal of forum surfing and ultimately was a bit frustrating. Initially I tried updating to the most recent version of OpenCV, currently at 4.0. I was unable to compile simpler-capture however, because the package -libopencv-nonfree couldn't be found. I tried recompiling OpenCV but was met with errors since Ubuntu 14.04 LTS has version 2.8 of cmake installed, and the recompilation required at least version 3.5.

I purged cmake from the system and installed cmake 3.8, but even with this version, I wasn't able to recompile. Cmake had installed to /usr/local/bin, and the build process was only looking in /usr/bin for cmake source. Adding symbolic links didn't solve this build issue. Eventually after speaking with the TAs, I learned that OpenCV versions 3 and 4 were problematic, and that I should downgrade to version 2.

I purged OpenCV and cmake, reinstalled cmake 2.8 and then downloaded, built, and installed OpenCV2.4.13. On my first try, simpler-capture built as shown here:

Running simpler-capture is shown below:



```
ubuntu@tegra-ubuntu:~/RTES_Sathak_Adrian/Part3/simpler-capture$ ls /dev/
autofs                  fuse          loop6            net                 nvhost-vi.1  sw_sync                   tty16  tty33  tty50  ttyS1     vcsa1
block                   gpu_freq_max  loop7            network_latency     nvhost-vic   tegra_audio_avpchannel    tty17  tty34  tty51  ttyS2     vcsa2
bus                     gpu_freq_min  loop-control     network_throughput  nvidiactl    tegra_avpchannel          tty18  tty35  tty52  ttyS3     vcsa3
camera.pcl              i2c-0         mapper           null                nvmap        tegra-crypto              tty19  tty36  tty53  ttyTHS0   vcsa4
char                    i2c-1         max_online_cpus  nvhost-as-gpu       port         tegra_dc_0                tty2   tty37  tty54  ttyTHS1   vcsa5
console                 i2c-2         mem              nvhost-ctrl         ppp          tegra_dc_1                tty20  tty38  tty55  ttyTHS2   vcsa6
constraint_cpu_freq     i2c-3         min_online_cpus  nvhost-ctrl-gpu     ptmx         tegra_dc_ctrl             tty21  tty39  tty56  uhid      vcsa7
constraint_gpu_freq     i2c-4         mipi-cal         nvhost-ctrl-isp     pts          tegra_dtv                 tty22  tty4   tty57  uinput    vga_arbiter
constraint_online_cpus  i2c-5         mmcblk0          nvhost-ctrl-isp.1   quadd        tegra-throughput          tty23  tty40  tty58  urandom   video0
core                    input         mmcblk0p1        nvhost-ctrl-vi      quadd_auth   timerinfo                 tty24  tty41  tty59  v4l       watchdog0
cpu_dma_latency         keychord      mmcblk0p2        nvhost-ctrl-vi.1    random       tty                       tty25  tty42  tty6   vcs       xt_qtaguid
cpu_freq_max            kmsg          mmcblk0p3        nvhost-dbg-gpu      rfkill       tty0                      tty26  tty43  tty60  vcs1      zero
cpu_freq_min            log           mmcblk0p4        nvhost-gpu          rtc          tty1                      tty27  tty44  tty61  vcs2      zram0
disk                    loop0         mmcblk0p5        nvhost-isp          rtc0         tty10                     tty28  tty45  tty62  vcs3
emc_freq_min            loop1         mmcblk0p6        nvhost-isp.1        shm          tty11                     tty29  tty46  tty63  vcs4
fb0                     loop2         mmcblk0p7        nvhost-msenc        snd          tty12                     tty3   tty47  tty7   vcs5
fb1                     loop3         mmcblk0p8        nvhost-prof-gpu     stderr       tty13                     tty30  tty48  tty8   vcs6
fd                      loop4         mmcblk0p9        nvhost-tsec         stdin        tty14                     tty31  tty49  tty9   vcs7
full                    loop5         mmcblk0rpmb      nvhost-vi           stdout       tty15                     tty32  tty5   ttyS0  vcsa
ubuntu@tegra-ubuntu:~/RTES_Sathak_Adrian/Part3/simpler-capture$ ./capture
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
```

Figure 7: Running simpler-capture

Part 4

- Choose a continuous transformation OpenCV example from computer-vision such as the canny-interactive, hough-interactive, hough-eliptical-interactive, or stereo-transform-impoved or the from the same 4 transforms in computer vision cv3 tested.   Show a screenshot to prove you built and ran the code. Provide a detailed explanation of the code and research uses for the continuous transformation by looking up API functions in the OpenCV manual (http://docs.opencv.org)

I compiled and ran the simple-canny-interactive, simple-hough-interactive, and simple-hough-elliptical-interactive programs. See the "compiling" screenshots in the Part4 folder of the submission.

The simple-hough-interactive program takes an image, finds the straight lines in that image, and then overlays markers of those straight lines into the display. Explanation of the simple-hough-interactive code upon entering main() follows.

Before the while(1) loop:

Firstly, a window is created for the camera's image will be displayed. Parameters are given such that the window is named "Capture Example" and the window is auto-sized to fit the size of the image. The dimensions that will be used for this are given as #defines, 640 x 480 pixels.

Instances of two classes (CvCapture and IplImage) are declared. The CvCapture class is used to facilitate capture of video from a camera. The IplImage class is from the Intel Image Processing Library, and is used to process a single frame. An int is created to identify the device as video"0", and then a few Mat (matrix) objects are declared. Finally a vector of type Vec4i is defined.

After command-line arguments are processed, the capture is initialized with a call to cvCreateCameraCapture() with the device identifier passed as the parameter. Then the height and width of the capture are set to the #defined dimensions for height and width resolutions with calls to cvSetCaptureProperty() and the #defines mentioned before.

Then begins the while(1) loop:

A frame (of type IiplImage) is captured with a call to cvQueryFrame() with the CvCapture class object passed as a parameter. This call gets the last frame from the CvCapture classes internal buffer, that's been filling since the cvCreateCameraCapture() call.

The frame is added to a new matrix "mat_frame". Then this matrix has an edge-detection function applied to it called Canny(). This takes the matrix which contains the last frame “mat_frame” and applies the transformation within the threshold limits [50, 200], outputting the result to the matrix canny_frame. This transformation is an edge detector, (this transformation alone can be seen in the simple-canny-interactive program) and would display edges as white lines on a black background.

This “canny_frame” matrix then passed to another transformation function: HoughLines(). This function takes the edge lines detected in canny_frame and tries to detect the straight lines among those edges. The call to HoughLines() in simple-hough-interactive uses 1 pixel and 1 degree for resolution, 50 intersections as the threshold for detection, and 50 and 10 for the distance resolutions of rho and theta respectively.

The resulting straight lines detected are put into a vector Vec4i. In a for-loop, each of these lines is drawn onto the matrix “mat_frame”, using the point vertices the color red (given as a RGB scalar), and a thickness of 3 pixels. Furthermore the lines are drawn as an antialiased line, which are specifically rendered for smoothness.

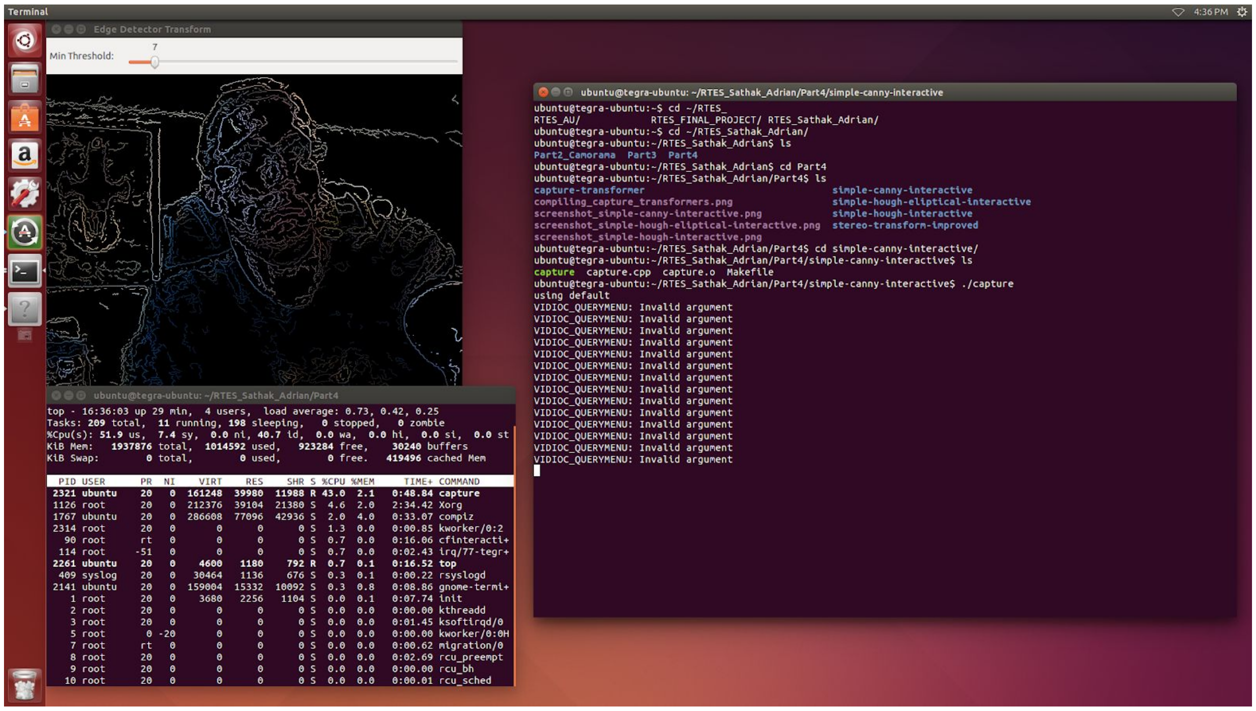
After the lines have been added to the matrix, and if the frame wasn’t null to begin with, the frame is shown to the “Capture Example” window. A cvWaitKey() call then waits for 10 milliseconds. If a key is detected and that key is “escape” (key 27), the while(1) loop ends and the program exits. Otherwise the while(1) loop continues, more frames are captured and transformed first to find edges and then straight lines, and finally displayed to the “Capture Example” window.

Upon detection of the “escape” key, the capture is ended with a call to cvReleaseCapture() and resources are freed. Then the “Capture Example” display window is closed with a call to cvDestroyWindow(), and the program exits.

<u>simple-hough-interactive uses</u>

The simple-hough-interactive continuous transform has many possible uses, particularly in research. For instance, airborne imagery of a forest with many Pine trees can be clouded with all the branches and foliage, but if you just want to find the tree trunks the HoughLine() transform performed in simple-hough-interactive would be a good place to start. Other types of images in which you might want to find straight lines could be satellite data in order to map roads for cartographic purposes, or sidescan sonar data in which you could be looking for wrecked ships and planes.

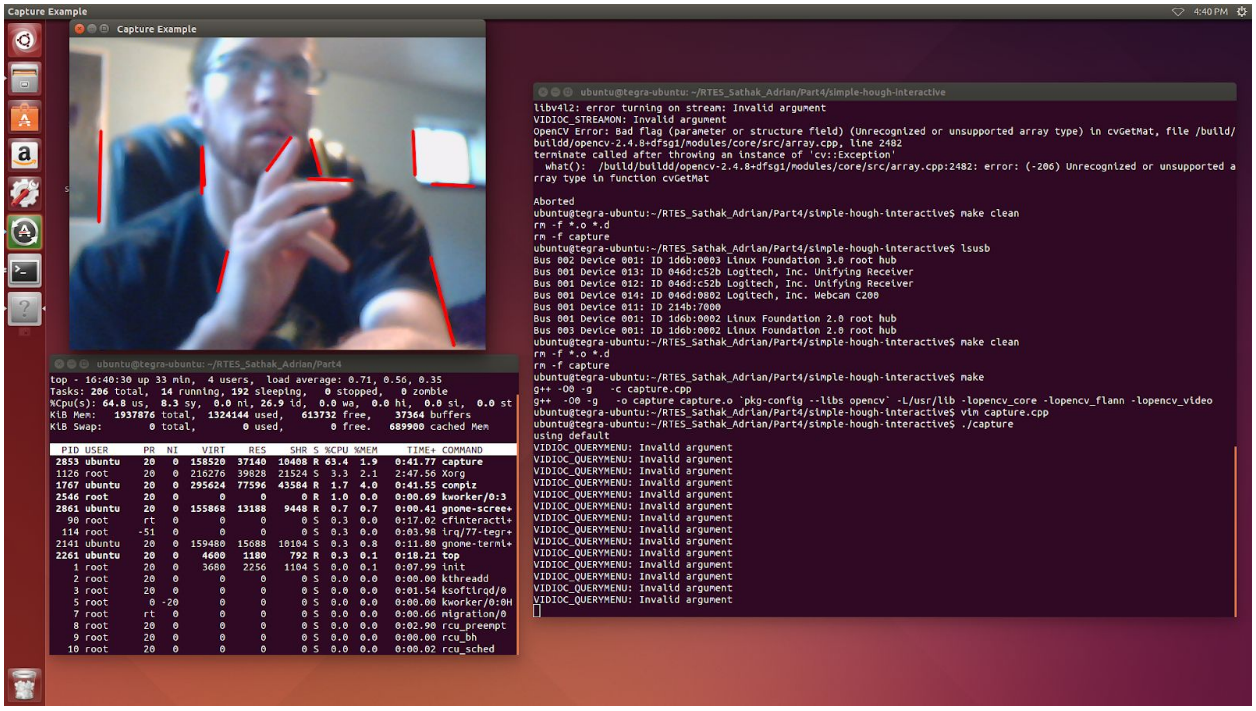
Screenshots of each of the transformations (simple-canny-interactive, simple-hough-interactive, and simple-hough-eliptical-interactive) follow. Additionally, note that I took screenshots with an open terminal window running the “top” command. This allows us to see the CPU usage of each program as they run.


Terminal
4:36 PM
Edge Detector Transform
Min Threshold:
7
ubuntu@tegra-ubuntu: ~/RTES_Sathak_Adrian/Part4/simple-canny-interactive
ubuntu@tegra-ubuntu: ~/RTES_Sathak_Adrian/Part4

```
ubuntu@tegra-ubuntu:~$ cd ~/RTES_
RTES_AU/            RTES_FINAL_PROJECT/ RTES_Sathak_Adrian/
ubuntu@tegra-ubuntu:~$ cd ~/RTES_Sathak_Adrian/
ubuntu@tegra-ubuntu:~/RTES_Sathak_Adrian$ ls
Part2_Camorama  Part3  Part4
ubuntu@tegra-ubuntu:~/RTES_Sathak_Adrian$ cd Part4
ubuntu@tegra-ubuntu:~/RTES_Sathak_Adrian/Part4$ ls
capture-transformer                                  simple-canny-interactive
compiling_capture_transformers.png                   simple-hough-eliptical-interactive
screenshot_simple-canny-interactive.png              simple-hough-interactive
screenshot_simple-hough-eliptical-interactive.png  stereo-transform-improved
screenshot_simple-hough-interactive.png
ubuntu@tegra-ubuntu:~/RTES_Sathak_Adrian/Part4$ cd simple-canny-interactive/
ubuntu@tegra-ubuntu:~/RTES_Sathak_Adrian/Part4/simple-canny-interactive$ ls
capture  capture.cpp  capture.o  Makefile
ubuntu@tegra-ubuntu:~/RTES_Sathak_Adrian/Part4/simple-canny-interactive$ ./capture
using default
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
```

```
top - 16:36:03 up 29 min,  4 users,  load average: 0.73, 0.42, 0.25
Tasks: 209 total,  11 running, 198 sleeping,   0 stopped,   0 zombie
%Cpu(s): 51.9 us,  7.4 sy,  0.0 ni, 40.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   1937876 total,  1014592 used,   923284 free,    30240 buffers
KiB Swap:        0 total,        0 used,        0 free.   419496 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 2321 ubuntu    20   0  161248  39980  11988 R 43.0  2.1   0:48.84 capture
 1126 root      20   0  212376  39104  21380 S  4.6  2.0   2:34.42 Xorg
 1767 ubuntu    20   0  286608  77096  42936 S  2.0  4.0   0:33.07 compiz
 2314 root      20   0       0      0      0 S  1.3  0.0   0:00.85 kworker/0:2
   90 root      rt   0       0      0      0 S  0.7  0.0   0:16.06 cfinteracti+
  114 root     -51   0       0      0      0 S  0.7  0.0   0:02.43 irq/77-tegr+
 2261 ubuntu    20   0    4600   1180    792 R  0.7  0.1   0:16.52 top
  409 syslog    20   0   30464   1136    676 S  0.3  0.1   0:00.22 rsyslogd
 2141 ubuntu    20   0  159004  15332  10092 S  0.3  0.8   0:08.86 gnome-termi+
    1 root      20   0    3680   2256   1104 S  0.0  0.1   0:07.74 init
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:01.45 ksoftirqd/0
    5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0H
    7 root      rt   0       0      0      0 S  0.0  0.0   0:00.62 migration/0
    8 root      20   0       0      0      0 S  0.0  0.0   0:02.69 rcu_preempt
    9 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_bh
   10 root      20   0       0      0      0 S  0.0  0.0   0:00.01 rcu_sched
```

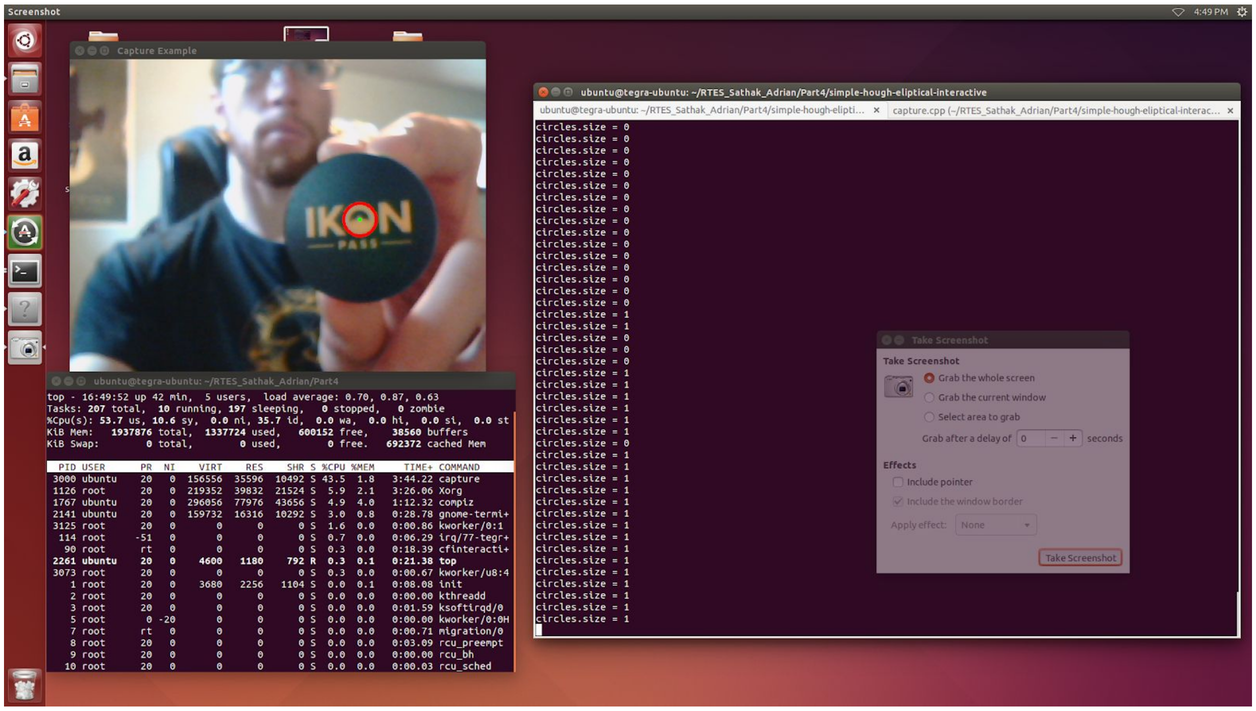
Figure 8: Running simple-canny-interactive


Capture Example
4:40 PM
Capture Example
ubuntu@tegra-ubuntu: ~/RTES_Sathak_Adrian/Part4/simple-hough-interactive
ubuntu@tegra-ubuntu: ~/RTES_Sathak_Adrian/Part4

```
libv4l2: error turning on stream: Invalid argument
VIDIOC_STREAMON: Invalid argument
OpenCV Error: Bad flag (parameter or structure field) (Unrecognized or unsupported array type) in cvGetMat, file /build/
buildd/opencv-2.4.8+dfsg1/modules/core/src/array.cpp, line 2482
terminate called after throwing an instance of 'cv::Exception'
  what():  /build/buildd/opencv-2.4.8+dfsg1/modules/core/src/array.cpp:2482: error: (-206) Unrecognized or unsupported a
rray type in function cvGetMat

Aborted
ubuntu@tegra-ubuntu:~/RTES_Sathak_Adrian/Part4/simple-hough-interactive$ make clean
rm -f *.o *.d
rm -f capture
ubuntu@tegra-ubuntu:~/RTES_Sathak_Adrian/Part4/simple-hough-interactive$ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 013: ID 046d:c52b Logitech, Inc. Unifying Receiver
Bus 001 Device 012: ID 046d:c52b Logitech, Inc. Unifying Receiver
Bus 001 Device 014: ID 046d:0802 Logitech, Inc. Webcam C200
Bus 001 Device 011: ID 214b:7000
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
ubuntu@tegra-ubuntu:~/RTES_Sathak_Adrian/Part4/simple-hough-interactive$ make clean
rm -f *.o *.d
rm -f capture
ubuntu@tegra-ubuntu:~/RTES_Sathak_Adrian/Part4/simple-hough-interactive$ 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
ubuntu@tegra-ubuntu:~/RTES_Sathak_Adrian/Part4/simple-hough-interactive$ vim capture.cpp
ubuntu@tegra-ubuntu:~/RTES_Sathak_Adrian/Part4/simple-hough-interactive$ ./capture
using default
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
```

```
top - 16:40:30 up 33 min,  4 users,  load average: 0.71, 0.56, 0.35
Tasks: 206 total,  14 running, 192 sleeping,   0 stopped,   0 zombie
%Cpu(s): 64.8 us,  8.3 sy,  0.0 ni, 26.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   1937876 total,  1324144 used,   613732 free,    37364 buffers
KiB Swap:        0 total,        0 used,        0 free.   689900 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 2853 ubuntu    20   0  158520  37140  10408 R 63.4  1.9   0:41.77 capture
 1126 root      20   0  216276  39828  21524 S  3.3  2.1   2:47.56 Xorg
 1767 ubuntu    20   0  295624  77596  43584 R  1.7  4.0   0:41.55 compiz
 2546 root      20   0       0      0      0 R  1.0  0.0   0:00.69 kworker/0:3
 2861 ubuntu    20   0  155868  13188   9448 R  0.7  0.7   0:00.41 gnome-scree+
   90 root      rt   0       0      0      0 S  0.3  0.0   0:17.02 cfinteracti+
  114 root     -51   0       0      0      0 S  0.3  0.0   0:03.98 irq/77-tegr+
 2141 ubuntu    20   0  159480  15688  10104 S  0.3  0.8   0:11.80 gnome-termi+
 2261 ubuntu    20   0    4600   1180    792 R  0.3  0.1   0:18.21 top
    1 root      20   0    3680   2256   1104 S  0.0  0.1   0:07.99 init
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:01.54 ksoftirqd/0
    5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0H
    7 root      rt   0       0      0      0 S  0.0  0.0   0:00.66 migration/0
    8 root      20   0       0      0      0 S  0.0  0.0   0:02.90 rcu_preempt
    9 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_bh
   10 root      20   0       0      0      0 S  0.0  0.0   0:00.02 rcu_sched
```

Figure 9: Running simple-hough-interactive



ubuntu@tegra-ubuntu: ~/RTES_Sathak_Adrian/Part4

```
top - 16:49:52 up 42 min,  5 users,  load average: 0.70, 0.87, 0.63
Tasks: 207 total,  10 running, 197 sleeping,   0 stopped,   0 zombie
%Cpu(s): 53.7 us, 10.6 sy,  0.0 ni, 35.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   1937876 total,  1337724 used,   600152 free,    38560 buffers
KiB Swap:        0 total,        0 used,        0 free.   692372 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 3000 ubuntu    20   0  156556  35596  10492 S 43.5  1.8   3:44.22 capture
 1126 root      20   0  219352  39832  21524 S  5.9  2.1   3:26.06 Xorg
 1767 ubuntu    20   0  296056  77976  43656 S  4.9  4.0   1:12.32 compiz
 2141 ubuntu    20   0  159732  16316  10292 S  3.0  0.8   0:28.78 gnome-termi+
 3125 root      20   0       0      0      0 S  1.6  0.0   0:00.86 kworker/0:1
  114 root     -51   0       0      0      0 S  0.7  0.0   0:06.29 irq/77-tegr+
   90 root      rt   0       0      0      0 S  0.3  0.0   0:18.39 cfinteracti+
 2261 ubuntu    20   0    4600   1180    792 R  0.3  0.1   0:21.38 top
 3073 root      20   0       0      0      0 S  0.3  0.0   0:00.67 kworker/u8:4
    1 root      20   0    3680   2256   1104 S  0.0  0.1   0:08.08 init
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:01.59 ksoftirqd/0
    5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0H
    7 root      rt   0       0      0      0 S  0.0  0.0   0:00.71 migration/0
    8 root      20   0       0      0      0 S  0.0  0.0   0:03.09 rcu_preempt
    9 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_bh
   10 root      20   0       0      0      0 S  0.0  0.0   0:00.03 rcu_sched
```

ubuntu@tegra-ubuntu: ~/RTES_Sathak_Adrian/Part4/simple-hough-eliptical-interactive

ubuntu@tegra-ubuntu: ~/RTES_Sathak_Adrian/Part4/simple-hough-elipti... × capture.cpp (~/RTES_Sathak_Adrian/Part4/simple-hough-eliptical-interac... ×

```
circles.size = 0
circles.size = 0
circles.size = 0
circles.size = 0
circles.size = 0
circles.size = 0
circles.size = 0
circles.size = 0
circles.size = 0
circles.size = 0
circles.size = 0
circles.size = 0
circles.size = 0
circles.size = 0
circles.size = 0
circles.size = 0
circles.size = 1
circles.size = 1
circles.size = 0
circles.size = 0
circles.size = 0
circles.size = 1
circles.size = 1
circles.size = 1
circles.size = 1
circles.size = 1
circles.size = 1
circles.size = 0
circles.size = 1
circles.size = 1
circles.size = 1
circles.size = 1
circles.size = 1
circles.size = 1
circles.size = 1
circles.size = 1
circles.size = 1
circles.size = 1
circles.size = 1
circles.size = 1
circles.size = 1
circles.size = 1
circles.size = 1
```

Figure 10: Running simple-hough-elliptical-interactive

- for stereo-transform-improved either implement or explain how you could make this work continuously rather than snapshot only.

After downloading the code from Prof. Siewert’s website, I was a bit confused by the task we were assigned. There are two programs included in the directory, capture.cpp and capture_stereo.cpp. The capture.cpp program already has a continuous stream capability, and the capture_stereo.cpp program also encapsulates the abilities to make continuous Canny (edge detection) and Hough (straight line) transformations within while(1) loops. What doesn’t exist is solely the ability to save images to the disc with these applied transformations.

It should be noted that files are only saved in capture_stereo.cpp when the “escape” key is pressed, in a snapshot fashion. I have doubts this was the intended update to the program, but images can be continuously written to the disc by removing the “if” condition that looks for “char c == ESC_KEY”. See capture_stereo.cpp in the stereo-transform-improved folder in the Part4 folder of the submission.

Part 5

Using a Logitech C200, choose 3 real-time interactive transformations to compare in terms of average frame rate at a given resolution for a range of at least 3 resolutions in a common aspect ratio (e.g. 4:3 for 1280x960, 640x480, 320x240, 160x120, 80x60) by adding time-stamps to analyze the potential throughput. Based on average analysis, pick a reasonable soft real-time deadline (e.g. if average frame rate is 12 Hz, choose a deadline of 100 milliseconds to provide some margin) and convert the processing to SCHED_FIFO and determine if you can meet deadlines with predictability and measure jitter in the frame rate relative to your deadline.

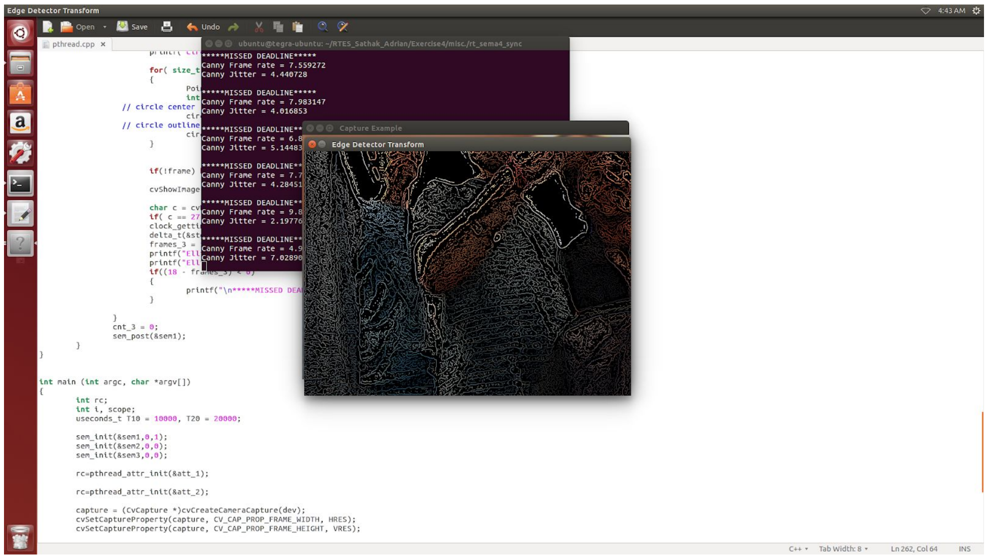
Ans:
Three threads have been created, with each thread running its own version of a transform, simple canny, hough and elliptical hough respectively.
The code has been designed such that three semaphores are used, one for each thread. Each thread has two while loops, one inner and one outer. The outer loop is a while(1), and the inner is a while loop for 50 frame counts.
The semaphore for thread 1(simple canny) has been initialized with value 1, and the other two semaphores with value 0. Each thread has a 'sem_wait' at the beginning, and releases the semaphore for the next thread at the end of its inner while loop. The chain continues so on and so forth allowing a type of FIFO scheduling using three semaphores' logic.
All screenshots below have been listed for 640 x 480 resolution.

The table below lists the FPS set for each transform per resolution:

Transform	640 x 480	320 x 240	160 x 120
Canny	10 fps	20 fps	25 fps
Hough	13 fps	32 fps	33 fps
Hough Elliptical	18 fps	20 fps	20 fps

The table below lists the deadline set for each transform per resolution:

Transform	640 x 480	320 x 240	160 x 120
Canny	125 ms	56 ms	45 ms
Hough	100 ms	33 ms	33 ms
Hough Elliptical	67 ms	56 ms	56 ms

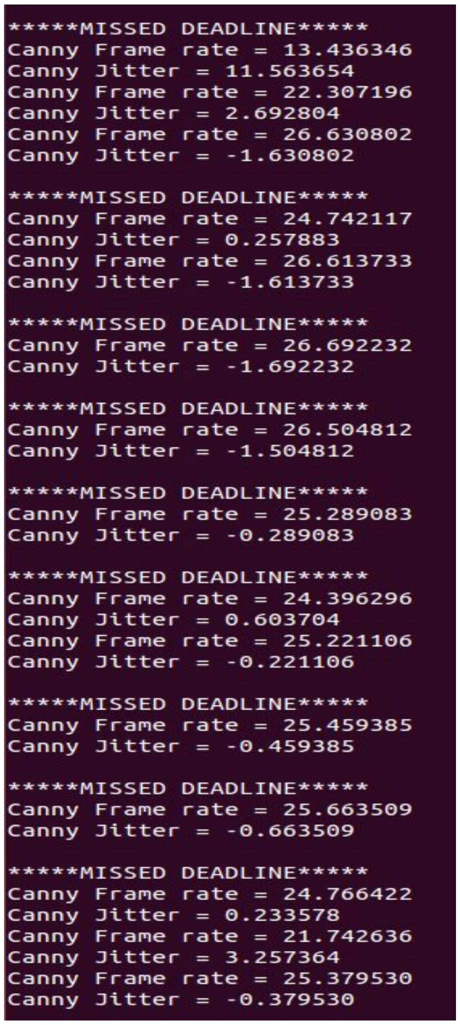


```
*****MISSED DEADLINE*****
Canny Frame rate = 7.559272
Canny Jitter = 4.440728

*****MISSED DEADLINE*****
Canny Frame rate = 7.983147
Canny Jitter = 4.016853

*****MISSED DEADLINE**
Canny Frame rate = 6.8
Canny Jitter = 5.14483

*****MISSED DEADLINE**
Canny Frame rate = 7.7
Canny Jitter = 4.28451

*****MISSED DEADLINE**
Canny Frame rate = 9.8
Canny Jitter = 2.19776

*****MISSED DEADLINE**
Canny Frame rate = 4.9
Canny Jitter = 7.02890
```

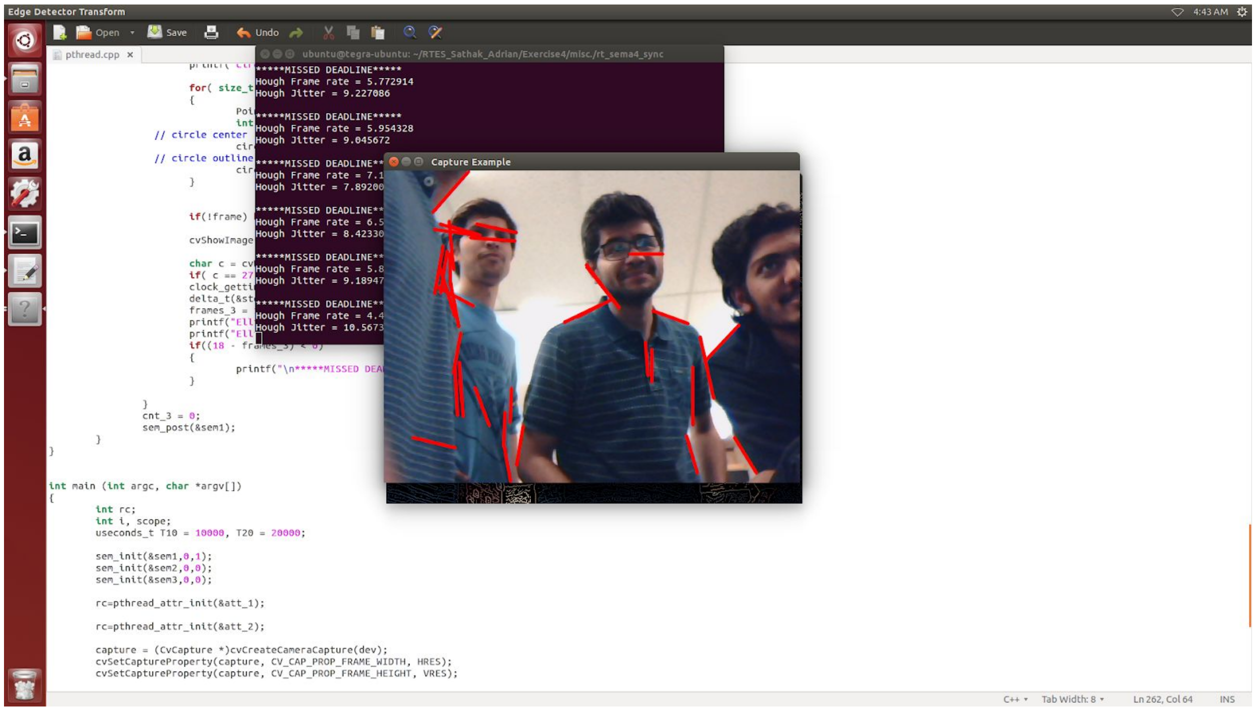
```
                for( size_t
                {
                        Poi
                        int
            // circle center
                        cir
            // circle outline
                        cir
                }


                if(!frame)

                cvShowImage

                char c = cv
                if( c == 27
                clock_getti
                delta_t(&st
                frames_3 =
                printf("Ell
                printf("Ell
                if((18 - frames_3) < 0)
                {
                        printf("\n*****MISSED DEA
                }

            }
            cnt_3 = 0;
            sem_post(&sem1);
        }
}


int main (int argc, char *argv[])
{
        int rc;
        int i, scope;
        useconds_t T10 = 10000, T20 = 20000;

        sem_init(&sem1,0,1);
        sem_init(&sem2,0,0);
        sem_init(&sem3,0,0);

        rc=pthread_attr_init(&att_1);

        rc=pthread_attr_init(&att_2);

        capture = (CvCapture *)cvCreateCameraCapture(dev);
        cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_WIDTH, HRES);
        cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_HEIGHT, VRES);
```

In the image above, one can see the canny transform of an image, with the output in the background, and a few deadlines missed.

As we can see in the screenshots below, some deadlines are missed occasionally.



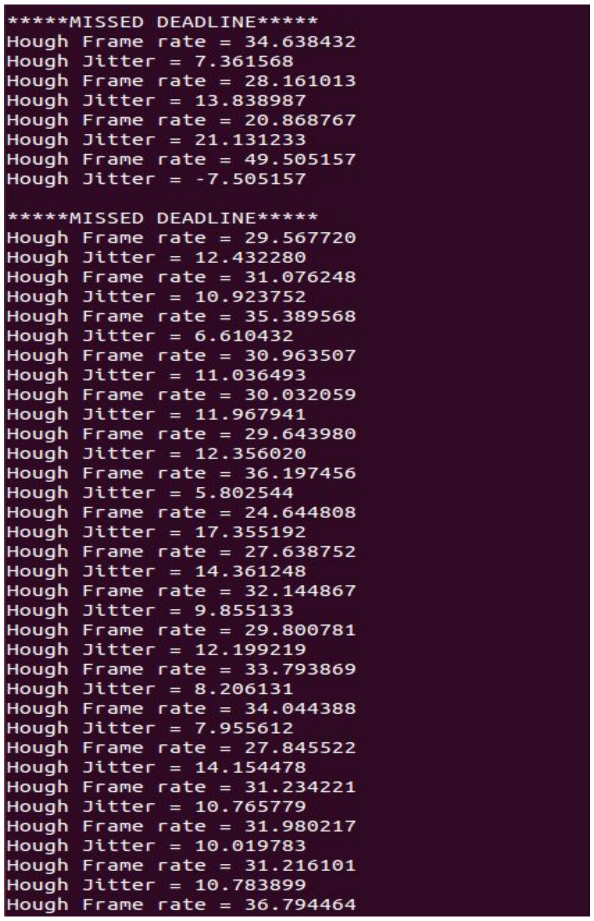
```
*****MISSED DEADLINE*****
Canny Frame rate = 13.436346
Canny Jitter = 11.563654
Canny Frame rate = 22.307196
Canny Jitter = 2.692804
Canny Frame rate = 26.630802
Canny Jitter = -1.630802

*****MISSED DEADLINE*****
Canny Frame rate = 24.742117
Canny Jitter = 0.257883
Canny Frame rate = 26.613733
Canny Jitter = -1.613733

*****MISSED DEADLINE*****
Canny Frame rate = 26.692232
Canny Jitter = -1.692232

*****MISSED DEADLINE*****
Canny Frame rate = 26.504812
Canny Jitter = -1.504812

*****MISSED DEADLINE*****
Canny Frame rate = 25.289083
Canny Jitter = -0.289083

*****MISSED DEADLINE*****
Canny Frame rate = 24.396296
Canny Jitter = 0.603704
Canny Frame rate = 25.221106
Canny Jitter = -0.221106

*****MISSED DEADLINE*****
Canny Frame rate = 25.459385
Canny Jitter = -0.459385

*****MISSED DEADLINE*****
Canny Frame rate = 25.663509
Canny Jitter = -0.663509

*****MISSED DEADLINE*****
Canny Frame rate = 24.766422
Canny Jitter = 0.233578
Canny Frame rate = 21.742636
Canny Jitter = 3.257364
Canny Frame rate = 25.379530
Canny Jitter = -0.379530
```



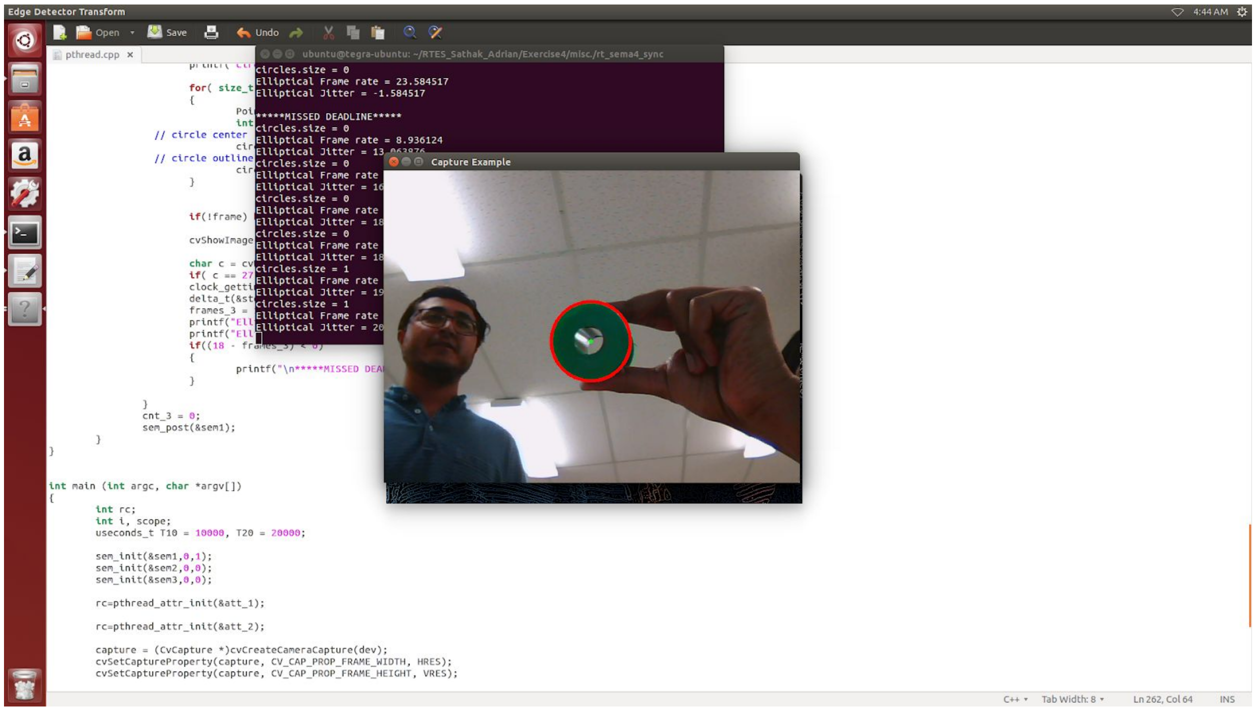
```
ubuntu@tegra-ubuntu: ~/RTES_Sathak_Adrian/Exercise4/misc./rt_sema4_sync
*****MISSED DEADLINE*****
Hough Frame rate = 5.772914
Hough Jitter = 9.227086

*****MISSED DEADLINE*****
Hough Frame rate = 5.954328
Hough Jitter = 9.045672

*****MISSED DEADLINE**
Hough Frame rate = 7.1
Hough Jitter = 7.89200

*****MISSED DEADLINE**
Hough Frame rate = 6.5
Hough Jitter = 8.42330

*****MISSED DEADLINE**
Hough Frame rate = 5.8
Hough Jitter = 9.18947

*****MISSED DEADLINE**
Hough Frame rate = 4.4
Hough Jitter = 10.5673
```

```
                for( size_t
                {
                        Poi
                        int
            // circle center
                        cir
            // circle outline
                        cir
                }


                if(!frame)

                cvShowImage

                char c = cv
                if( c == 27
                clock_getti
                delta_t(&st
                frames_3 =
                printf("Ell
                printf("Ell
                if((18 - frames_3) < 0)
                {
                        printf("\n*****MISSED DEA
                }

        }
        cnt_3 = 0;
        sem_post(&sem1);
    }
}


int main (int argc, char *argv[])
{
        int rc;
        int i, scope;
        useconds_t T10 = 10000, T20 = 20000;

        sem_init(&sem1,0,1);
        sem_init(&sem2,0,0);
        sem_init(&sem3,0,0);

        rc=pthread_attr_init(&att_1);

        rc=pthread_attr_init(&att_2);

        capture = (CvCapture *)cvCreateCameraCapture(dev);
        cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_WIDTH, HRES);
        cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_HEIGHT, VRES);
```

In the image above, one can see the hough line transform of an image, with the output in the background, and a few deadlines missed.



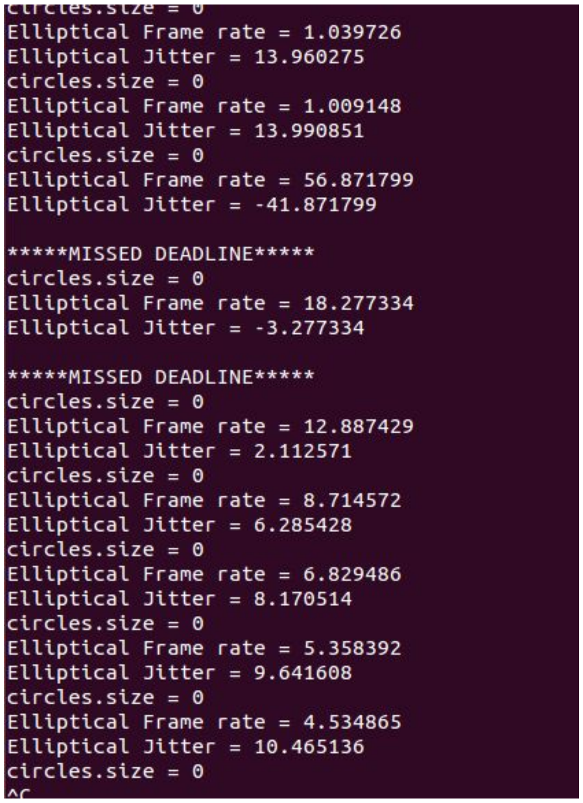
```
*****MISSED DEADLINE*****
Hough Frame rate = 34.638432
Hough Jitter = 7.361568
Hough Frame rate = 28.161013
Hough Jitter = 13.838987
Hough Frame rate = 20.868767
Hough Jitter = 21.131233
Hough Frame rate = 49.505157
Hough Jitter = -7.505157

*****MISSED DEADLINE*****
Hough Frame rate = 29.567720
Hough Jitter = 12.432280
Hough Frame rate = 31.076248
Hough Jitter = 10.923752
Hough Frame rate = 35.389568
Hough Jitter = 6.610432
Hough Frame rate = 30.963507
Hough Jitter = 11.036493
Hough Frame rate = 30.032059
Hough Jitter = 11.967941
Hough Frame rate = 29.643980
Hough Jitter = 12.356020
Hough Frame rate = 36.197456
Hough Jitter = 5.802544
Hough Frame rate = 24.644808
Hough Jitter = 17.355192
Hough Frame rate = 27.638752
Hough Jitter = 14.361248
Hough Frame rate = 32.144867
Hough Jitter = 9.855133
Hough Frame rate = 29.800781
Hough Jitter = 12.199219
Hough Frame rate = 33.793869
Hough Jitter = 8.206131
Hough Frame rate = 34.044388
Hough Jitter = 7.955612
Hough Frame rate = 27.845522
Hough Jitter = 14.154478
Hough Frame rate = 31.234221
Hough Jitter = 10.765779
Hough Frame rate = 31.980217
Hough Jitter = 10.019783
Hough Frame rate = 31.216101
Hough Jitter = 10.783899
Hough Frame rate = 36.794464
```



```
ubuntu@tegra-ubuntu: ~/RTES_Sathak_Adrian/Exercise4/misc./rt_sema4_sync
circles.size = 0
Elliptical Frame rate = 23.584517
Elliptical Jitter = -1.584517

*****MISSED DEADLINE*****
circles.size = 0
Elliptical Frame rate = 8.936124
Elliptical Jitter = 13.063876
circles.size = 0
Elliptical Frame rate
Elliptical Jitter = 16
circles.size = 0
Elliptical Frame rate
Elliptical Jitter = 18
circles.size = 0
Elliptical Frame rate
Elliptical Jitter = 18
circles.size = 1
Elliptical Frame rate
Elliptical Jitter = 19
circles.size = 1
Elliptical Frame rate
Elliptical Jitter = 20
```

```
                        for( size_t
                        {
                                Poi
                                int
                // circle center
                                cir
                // circle outline
                                cir
                        }


                        if(!frame)

                        cvShowImage

                        char c = cv
                        if( c == 27
                        clock_getti
                        delta_t(&st
                        frames_3 =
                        printf("Ell
                        printf("Ell
                        if((18 - frames_3) < 0)
                        {
                                printf("\n*****MISSED DEA
                        }

                }
                cnt_3 = 0;
                sem_post(&sem1);
        }
}


int main (int argc, char *argv[])
{
        int rc;
        int i, scope;
        useconds_t T10 = 10000, T20 = 20000;

        sem_init(&sem1,0,1);
        sem_init(&sem2,0,0);
        sem_init(&sem3,0,0);

        rc=pthread_attr_init(&att_1);

        rc=pthread_attr_init(&att_2);

        capture = (CvCapture *)cvCreateCameraCapture(dev);
        cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_WIDTH, HRES);
        cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_HEIGHT, VRES);
```

In the image above, one can see the hough elliptical transform of an image, with the output in the background, and a few deadlines missed.



```
circles.size = 0
Elliptical Frame rate = 1.039726
Elliptical Jitter = 13.960275
circles.size = 0
Elliptical Frame rate = 1.009148
Elliptical Jitter = 13.990851
circles.size = 0
Elliptical Frame rate = 56.871799
Elliptical Jitter = -41.871799

*****MISSED DEADLINE*****
circles.size = 0
Elliptical Frame rate = 18.277334
Elliptical Jitter = -3.277334

*****MISSED DEADLINE*****
circles.size = 0
Elliptical Frame rate = 12.887429
Elliptical Jitter = 2.112571
circles.size = 0
Elliptical Frame rate = 8.714572
Elliptical Jitter = 6.285428
circles.size = 0
Elliptical Frame rate = 6.829486
Elliptical Jitter = 8.170514
circles.size = 0
Elliptical Frame rate = 5.358392
Elliptical Jitter = 9.641608
circles.size = 0
Elliptical Frame rate = 4.534865
Elliptical Jitter = 10.465136
circles.size = 0
^C
```

The average frame rate for each resolution and transform has been observed, and an arbitrarily ranged value has been selected for calculating the deadline. If a deadline is missed, a “MISSED DEADLINE” tag is printed, and the relative jitter is calculated and displayed.

The final relative jitter can be predicted to a particular extent, and it is partly depends on the code as well the other processes running on the board at the time of code execution.
It was observed that even if a short gap is kept in between executing the code consecutively, the jitter does vary a bit, leading me to believe that the board has a number of other subsidiary processes running as well.