Department of Electrical and Computer Engineering

University of Colorado at Boulder

ECEN5623 - Real Time Embedded Systems

Exercise 4

Submitted by

Parth T — Parth K

Submitted on March 25, 2024

Contents

List of Figures

List of Tables

PDF is clickable

1 Question 1

Q: [10 points] Obtain a Logitech C200 or C270 camera or equivalent and verify that is detected by the DE1-SoC, Raspberry Pi or Jetson Board USB driver. You can check the camera out from eStore or purchase one of your own, or use another camera that has a compliant UVC driver. Use lsusb, lsmod and dmesg kernel driver configuration tool to make sure your Logitech C2xx USB camera is plugged in and recognized by your DE1-SoC, Raspberry Pi or Jetson (note that on the Jetson, it does not use driver modules, but rather a monolithic kernel image, so lsmod will not look the same as other Linux systems – see what you can find by exploring /cat/proc on you Jetson to find the camera USB device). For the Jetson, do lsusb — grep C200 (or C270) and prove to the TA (and more importantly yourself) with that output (screenshot) that your camera is recognized. For systems other than a Jetson, do lsmod — grep video and verify that the UVC driver is loaded as well (http://www.ideasonboard.org/uvc/). To further verify, or debug if you don't see the UVC driver loaded in response to plugging in the USB camera, do dmesg — grep video or just dmesg and scroll through the log messages to see if your USB device was found. Capture all output and annotate what you see with descriptions to the best of your understanding.

Answer: Verify Camera Detection

(a) Check USB Devices:

Command: lsusb
This command lists all USB devices currently connected to the system. Running this in the terminal helps you identify whether the Logitech camera is physically recognized by the system. The output includes details like the device ID and the manufacturer, which can be used to confirm the presence of the camera.

```
[..ta/university/RTES_ECEN5623/Assignments/Excercise_4/answers/q4]
✘ parth lsusb | grep C270
Bus 003 Device 022: ID 046d:0825 Logitech, Inc. Webcam C270
```

Figure 1: lsusb

(b) Verify UVC Driver Load:

Command: lsmod — grep video
Purpose: The lsmod command displays a list of loaded kernel modules, and piping this output to grep video filters the list to show only modules related to video, such as the UVC (USB Video Class) driver, often indicated by uvcvideo. This command verifies that the necessary driver for operating the camera is loaded in the system. The UVC driver is crucial because it provides generic support for USB video devices, ensuring that compliant cameras can work with a broad set of operating systems and applications without requiring proprietary drivers.

```
parth lsmod | grep uvcvideo
uvcvideo              135168  0
videobuf2_vmalloc      20480  1 uvcvideo
uvc                    12288  1 uvcvideo
videobuf2_v4l2         40960  1 uvcvideo
videodev              364544  2 videobuf2_v4l2,uvcvideo
videobuf2_common       86016  4 videobuf2_vmalloc,videobuf2_v4l2,uvcvideo,videobuf2_memops
mc                     86016  5 videodev,snd_usb_audio,videobuf2_v4l2,uvcvideo,videobuf2_common
```

Figure 2: lsmod

(c) Inspect Kernel Messages:
Commands: dmesg — grep video and dmesg
Purpose: These commands are used to inspect kernel messages. dmesg displays a log of system messages, including hardware detection, drivers loaded, and errors encountered during the boot process or while the system is running. dmesg — grep video narrows down this log to entries related to video devices and drivers. This can reveal whether the camera was successfully initialized, recognized, and configured by the kernel, or if there were any issues during its detection process.

```
parth  sudo dmesg | tail -f
[31994.011447] r8152 5-2.3:1.0 enx00e04c3606f4: renamed from eth0
[31994.100320] kauditd_printk_skb: 14 callbacks suppressed
[31994.100323] audit: type=1107 audit(1711329632.071:1166): pid=852 uid=102 auid=4294967295 ses=
4294967295 subj=unconfined msg='apparmor="DENIED" operation="dbus_signal"  bus="system" path="/o
rg/freedesktop/NetworkManager/Settings" interface="org.freedesktop.NetworkManager.Settings" memb
er="NewConnection" name=":1.3" mask="receive" pid=28429 label="snap.zoom-client.zoom-client" pee
r_pid=857 peer_label="unconfined"
               exe="/usr/bin/dbus-daemon" sauid=102 hostname=? addr=? terminal=?'
[32300.540322] usb 3-2: USB disconnect, device number 21
[32303.416323] usb 3-2: new high-speed USB device number 22 using xhci_hcd
[32303.784761] usb 3-2: New USB device found, idVendor=046d, idProduct=0825, bcdDevice= 0.12
[32303.784777] usb 3-2: New USB device strings: Mfr=0, Product=0, SerialNumber=2
[32303.784784] usb 3-2: SerialNumber: AB20F2B0
[32303.848058] usb 3-2: Found UVC 1.00 device <unnamed> (046d:0825)
```

Figure 3: Kernel Message

2 Question 2

Q: Option 2: Cheese If you do not have cheese, do sudo apt-get install cheese on your Jetson, Raspberry Pi or DE1-SoC board or other native Linux system. This should not only install nice camera capture GUI tools, but also the V4L2 API described well in this series of Linux articles - http://lwn.net/Articles/203924/ . Running cheese should provide an interactive camera control session for your Logitech C2xx camera – if you have issues connecting to your camera do a "man cheese" and specify your camera device file entry point (e.g. /dev/video0). Show that you tested your camera with a cheese screen dump and test photo.

Answer: Install Cheese: Execute the following command:

```
1 sudo apt-get install cheese
2
```

```
[..ta/university/RTES_ECEN5623/Assignments/Excercise_4/answers/q4]
parth  cheese --version
Cheese 41.1
```

Figure 4: Cheese Version

Manual Device Selection: If Cheese does not automatically detect your camera, or if you encounter issues, you can specify the camera device file manually:

Determine your camera's device file (/dev/video0 for the first detected video device). You can list video devices with ls /dev/video*. Launch Cheese with a specified device file by running: cheese –device=/dev/video0 (replace /dev/video0 with camera's device file if different).

Screen Dump: screenshot of the Cheese application window showing the live camera feed with keyboard shortcut (e.g., PrtScn key).

Figure 5: Cheese Output

3 Question 3

Q: [10 points] Using your verified Logitech C2xx camera on a DE1-SoC, Raspberry Pi or Jetson, verify that it can stream continuously to a raw image buffer for transformation and processing using any one of the below methods:

Answer:

(a) Update Package Index and Install Prerequisites: This step involves updating the package index on your Linux system and installing the minimal prerequisites required to compile OpenCV, such as cmake, g++, wget, and unzip.

(b) Download OpenCV Sources: The OpenCV source code is downloaded from its GitHub repository. The wget command is used to fetch the zip archive of the OpenCV source code designated by the 4.x version tag, which is then unpacked using unzip.

(c) Create a Build Directory: A separate build directory is created to keep the compilation process organized and separate from the source code.

(d) Configure the Build: The cmake command is used to configure the build environment. This step prepares the makefile based on the available source code and specified options.

(e) Compile OpenCV: The compilation process is initiated with the cmake –build . command. This step compiles the source code into executable binaries and libraries.

```
# Install minimal prerequisites (Ubuntu 18.04 as reference)
sudo apt update && sudo apt install -y cmake g++ wget unzip

# Download and unpack sources
wget -O opencv.zip https://github.com/opencv/opencv/archive/4.x.zip
unzip opencv.zip

# Create build directory
mkdir -p build && cd build

# Configure
cmake ../opencv-4.x

# Build
cmake --build .

# make
make -j4

# make install
sudo make install

```

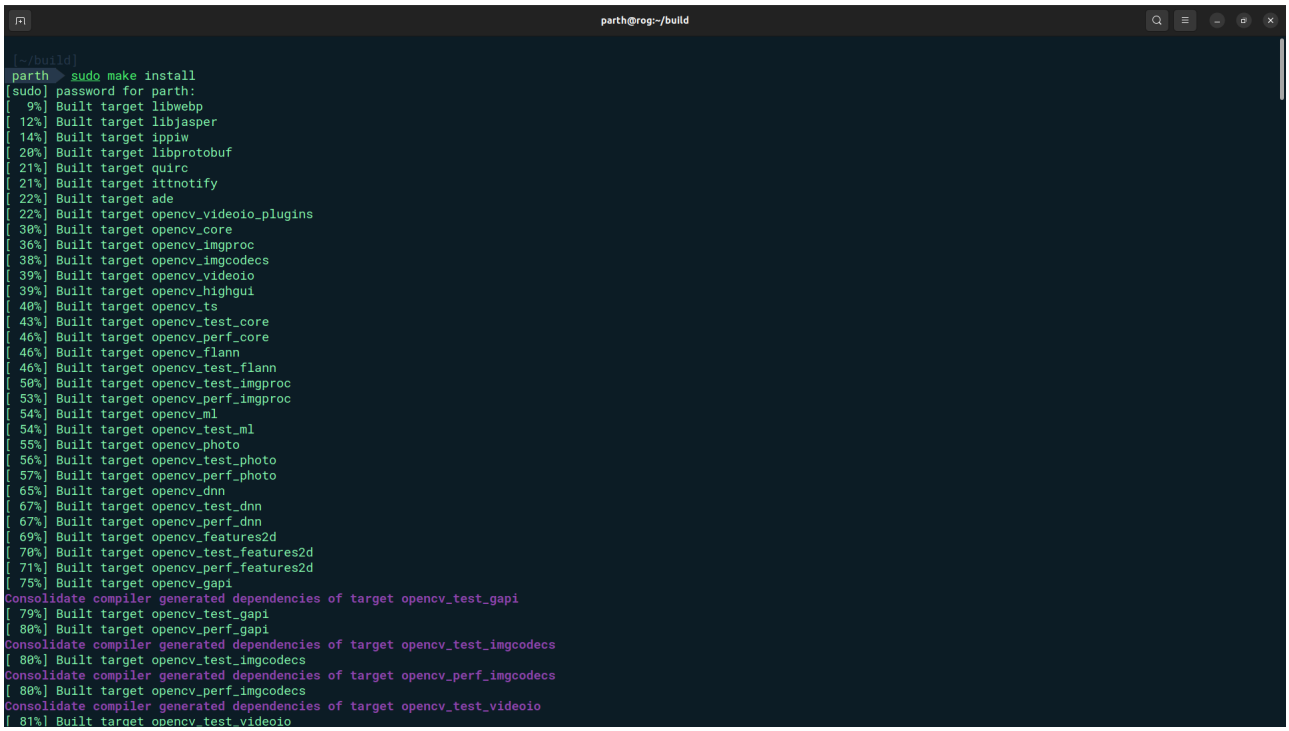


Figure 6: Build 1

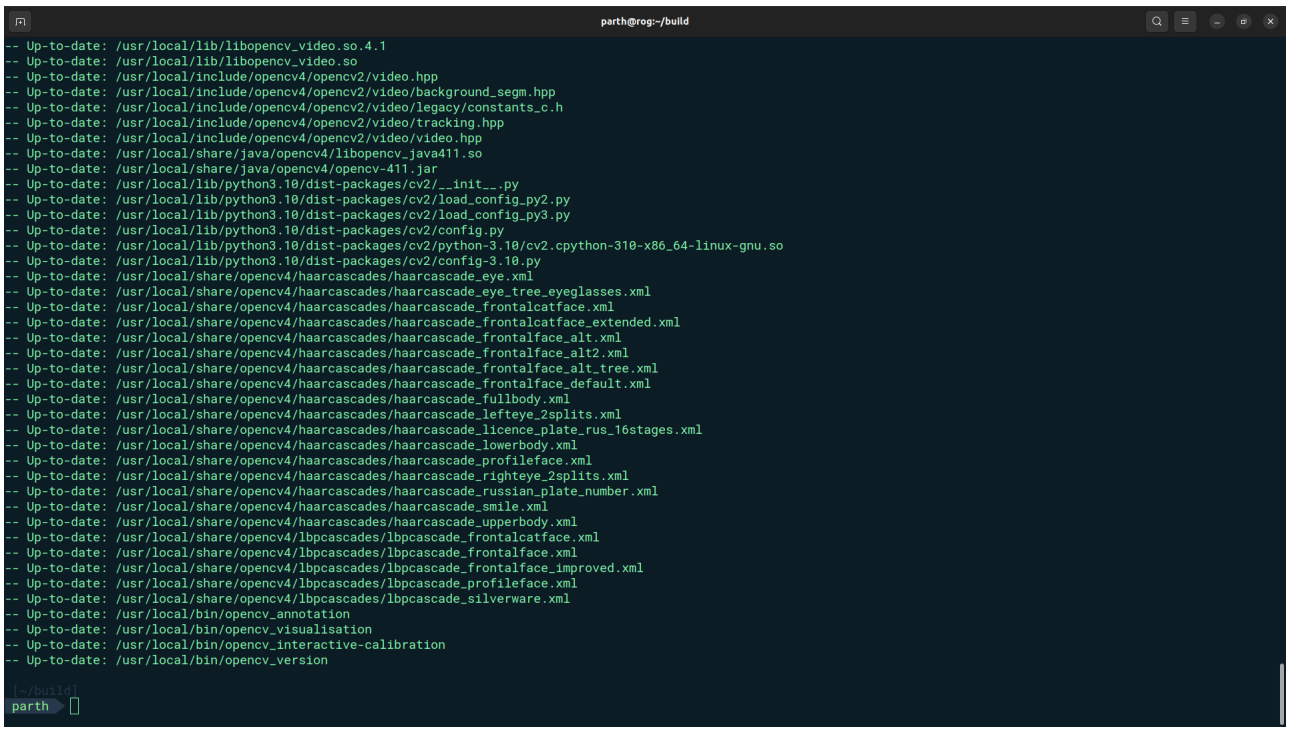


Figure 7: Build 2

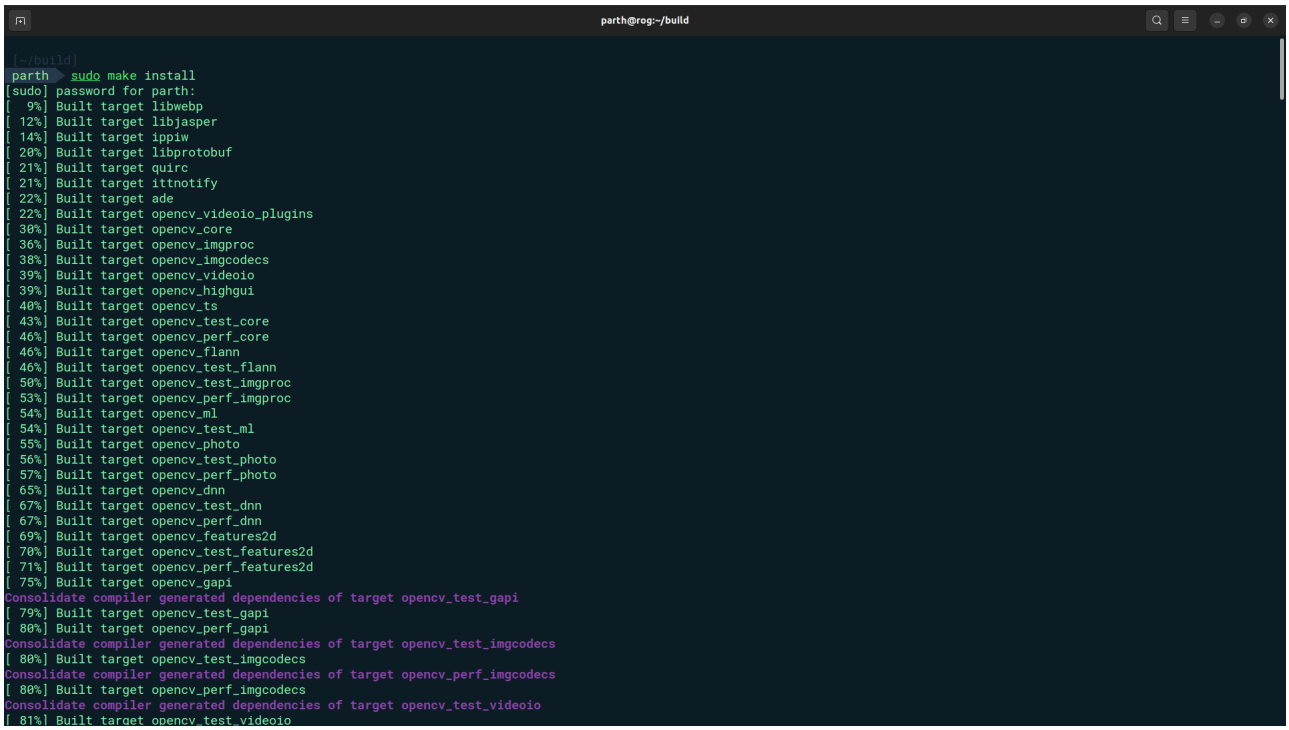


Figure 8: Build 3

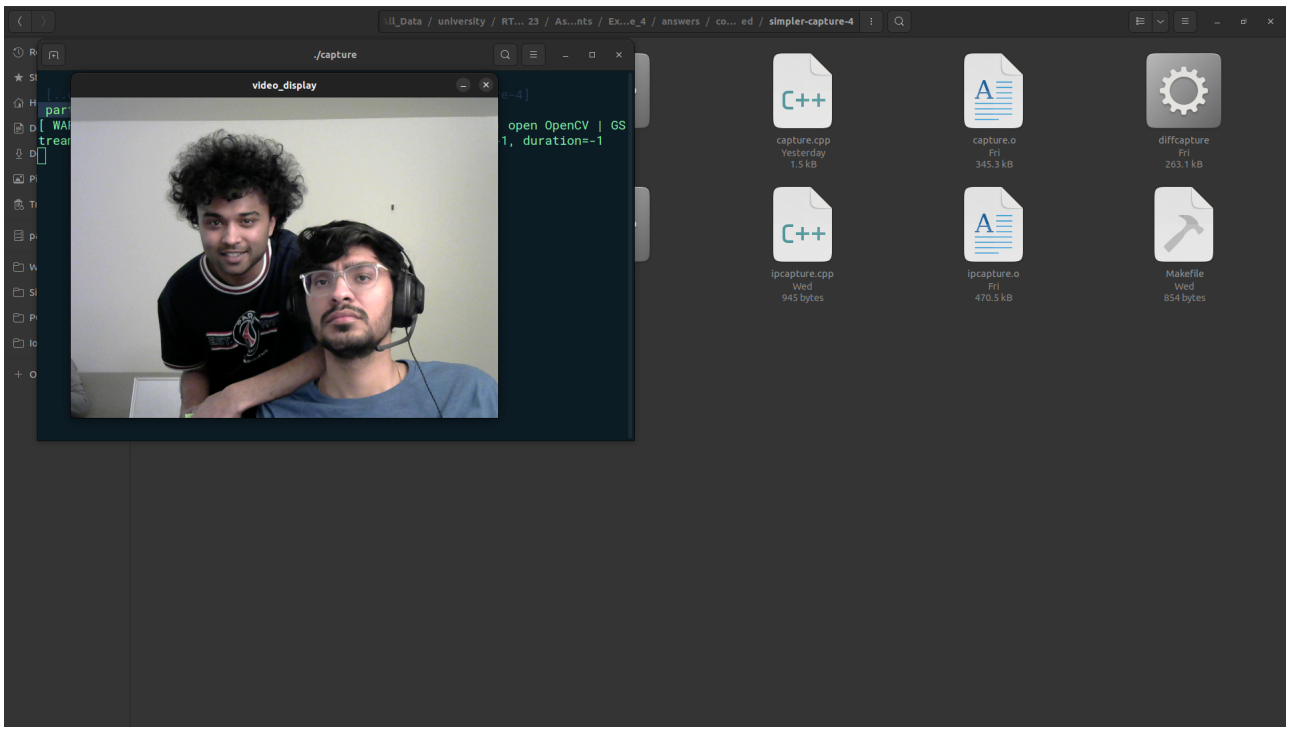


Figure 9: Build 4

(a) VideoCapture cam0(0);: Creates an object cam0 for capturing video from the camera. The parameter (0) specifies the first camera device. This class provides a way to interact with the video capturing functionality of OpenCV.

(b) namedWindow"$video_display$";: Creates a window named "video_display" for displaying the video. This function is part of the High-level GUI module.

(c) cam0.isOpened(): Checks if the video capture device has been successfully initialized.

(d) exit(SYSTEM_ERROR);: Exits the program with a system-defined error status if the camera cannot be opened.

(e) cam0.set$CAP_PROP_FRAME_WIDTH, 640$; and cam0.set$CAP_PROP_FRAME_HEIGHT, 480$;: Sets the properties of the capture device, specifically the frame width and height to 640x480 pixels. CAP_PROP_FRAME_WIDTH and CAP_PROP_FRAME_HEIGHT are predefined constants that specify which property to set.

(f) Mat frame$640, 480$,;: Defines a matrix (Mat) object named frame to store individual frames captured from the camera. The constructor parameters typically include the dimensions and the type of the matrix, but here it seems to be a typo or an incomplete line of code, as the type is missing.

(g) cam0.read$frame$;: Captures the next frame from the video capture device and stores it in frame. This method returns false if no frames have been captured.

(h) imshow"$video\_display$", $frame$;: Displays the frame in the window named "video_display". This function is called within the loop to continuously update the window with new frames from the camera.

(i) waitKey(10): Waits for a key event for a brief period (10 milliseconds). This function is essential for the GUI to process events and update the window. The return value is stored in winInput.

(j) if ((winInput = waitKey(10)) == ESCAPE_KEY): Checks if the ESC key $ASCII value 27$ has been pressed. If so, it breaks out of the loop, ending the video capture.

(k) destroyWindow"$video_display$";: Closes the window named "video_display" before the program exits.

4 Question 4

Q: [15 points] Review POSIX-examples and especially POSIX_MQ_LOOP and build the code related to POSIX message queues and run them to learn about basic use.

Answer:

(a)

Canny Here's an explanation of the provided Code

- Mat variables canny_frame, timg_gray, timg_grad, and frame are declared to store image data.
- Integer variables lowThreshold, max_lowThreshold, ratio, and kernel_size are declared for Canny edge detection parameters.

CannyThreshold Function:

- This function is used to perform Canny edge detection on the captured frame.
- It converts the frame to grayscale using cvtColor().
- Noise reduction is applied using blur().
- Canny edge detection is performed using Canny().
- The result is displayed in the window_name window using imshow().

Main Function:

- Video capture object cam0 is initialized for the default camera (index 0).
- A window named "video_display" is created for displaying the captured video frames.
- The frame size is set to 640x480 pixels using cam0.set().
- Within a while loop:
- Frames are continuously captured from the camera using cam0.read().
- The captured frame is displayed in the "video_display" window using imshow().
- The Canny edge detection trackbar is created and CannyThreshold function is called to apply edge detection on the frame.
- The program exits the loop if the escape key (ESC) is pressed.

This code captures video frames from the camera, performs Canny edge detection, and

Cannycam

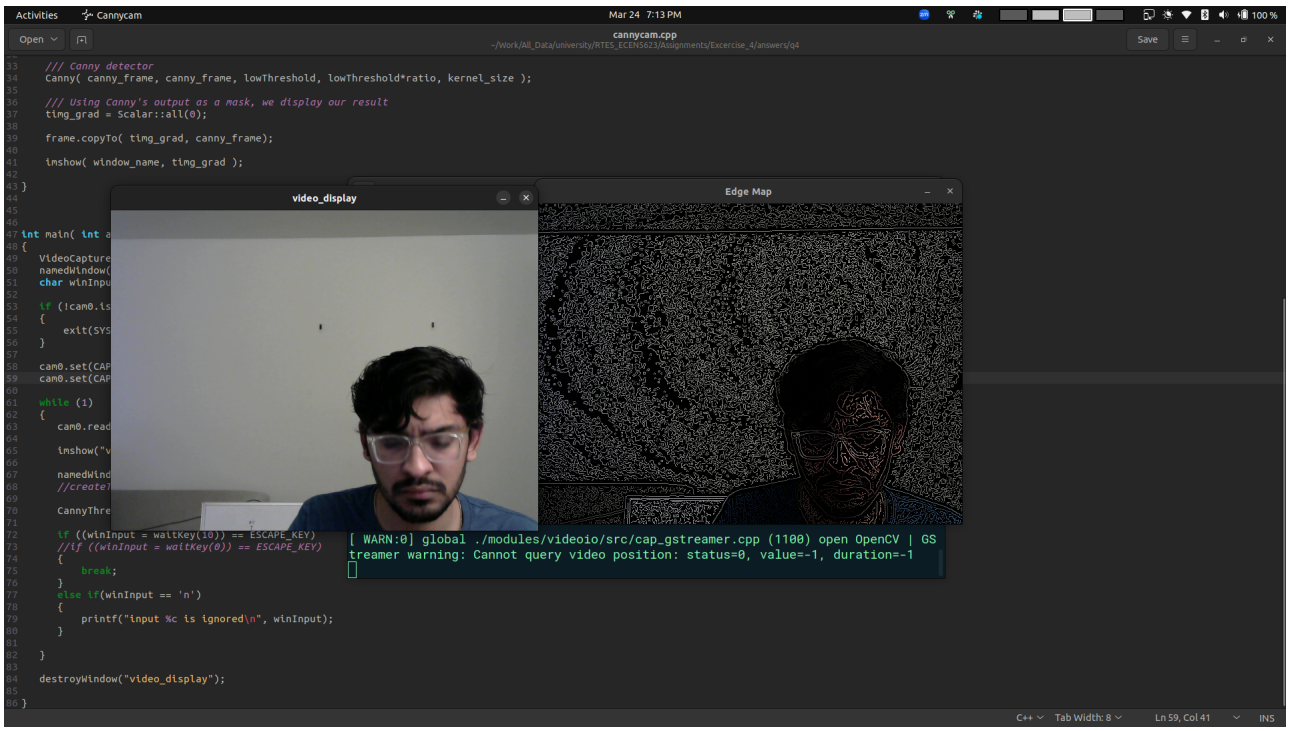
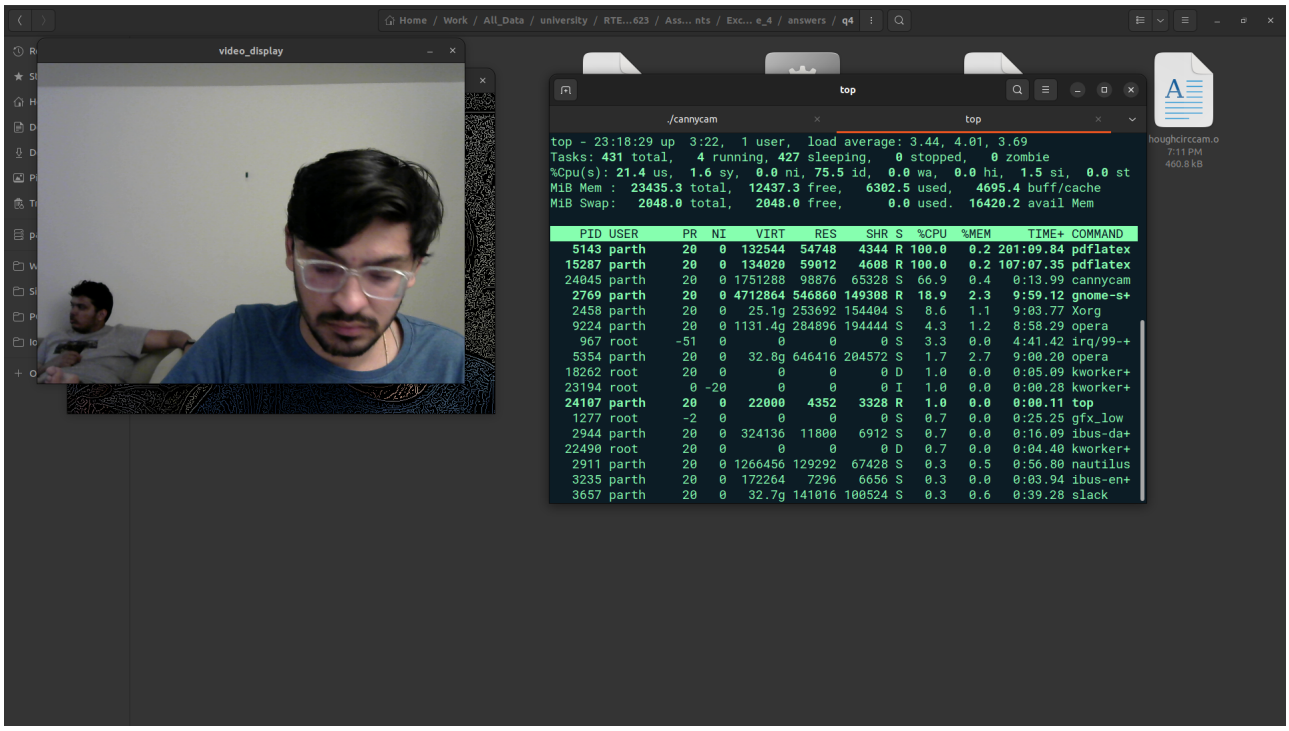


Figure 10: Canny

Cannycam top



PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
5143	parth	20	0	132544	54748	4344	R	100.0	0.2	201:09.84	pdflatex
15287	parth	20	0	134020	59012	4608	R	100.0	0.2	107:07.35	pdflatex
24045	parth	20	0	1751288	98876	65328	S	66.9	0.4	0:13.99	cannycam
2769	parth	20	0	4712864	546860	149308	R	18.9	2.3	9:59.12	gnome-s+
2458	parth	20	0	25.1g	253692	154404	S	8.6	1.1	9:03.77	Xorg
9224	parth	20	0	1131.4g	284896	194444	S	4.3	1.2	8:58.29	opera
967	root	-51	0	0	0	0	S	3.3	0.0	4:41.42	irq/99-+
5354	parth	20	0	32.8g	646416	204572	S	1.7	2.7	9:00.20	opera
18262	root	20	0	0	0	0	D	1.0	0.0	0:05.09	kworker+
23194	root	0	-20	0	0	0	I	1.0	0.0	0:00.28	kworker+
24107	parth	20	0	22000	4352	3328	R	1.0	0.0	0:00.11	top
1277	root	-2	0	0	0	0	S	0.7	0.0	0:25.25	gfx_low
2944	parth	20	0	324136	11800	6912	S	0.7	0.0	0:16.09	ibus-da+
22490	root	20	0	0	0	0	D	0.7	0.0	0:04.40	kworker+
2911	parth	20	0	1266456	129292	67428	S	0.3	0.5	0:56.80	nautilus
3235	parth	20	0	172264	7296	6656	S	0.3	0.0	0:03.94	ibus-en+
3657	parth	20	0	32.7g	141016	100524	S	0.3	0.6	0:39.28	slack

Figure 11: Canny Top

Here's an explanation of the provided code

- Mat variables dst, cdst, and cdstP are declared to store images. default_file and filename are declared to store the default image file path and the actual image file path, respectively. src is declared to store the loaded image.

- Image Loading: imread() function is used to load an image from the specified file path (filename) in grayscale mode (IMREAD_GRAYSCALE).
- Edge Detection (Canny): Canny() function is applied to detect edges in the loaded image (src). The detected edges are stored in the dst matrix.
- Color Conversion: cvtColor() function converts the grayscale edge image (dst) to a BGR (Blue-Green-Red) color image (cdst) for visualization.
- Standard Hough Line Transform: HoughLines() function detects lines in the edge image (dst) using the Standard Hough
- Drawing Detected Lines: For each detected line, line() function draws a line on the output image (cdst) using the line parameters (rho and theta). Probabilistic Hough Line Transform: HoughLinesP() function detects lines in the edge image (dst) using the Probabilistic Hough Line Transform. Detected lines are stored in the linesP vector.
- Drawing Probabilistic Detected Lines: For each detected line, line() function draws a line on the output image (cdstP) using the line parameters (x1, y1, x2, y2). Additionally, lines are drawn on the original image (src) to visualize the detected lines.
- Displaying Results: imshow() function displays the original image (src) and the image with detected lines (cdstP) in separate windows. This code loads an image, performs edge detection using the Canny algorithm, detects lines in the edge image using both the Standard Hough Line Transform and the Probabilistic Hough Line Transform, and visualizes the detected lines on the original image. Finally, it displays the results in separate windows and waits for a key press before exiting.

Hough Line Transform

The following changes have been made to the houghline.cpp code to make it work with continuous video input in houghlinecam.cpp:

- Video Capture: The houghlinecam.cpp code includes the opencv2/videoio.hpp header file to access the video capture functionality. It creates a VideoCapture object cam0 to read frames from the default camera (index 0).
- Continuous Video Processing Loop: The code enters a continuous loop where it reads frames from the camera using cam0.read(frame), displays the frame in the "video_display" window using imshow("video_display", frame), performs edge detection using the Canny algorithm, copies the edges to the output images (cdst and cdstP), and calls the HoughLinePTransform() function to detect and draw lines.
- Keyboard Input Handling: The loop checks for keyboard input using waitKey(10), which waits for 10 milliseconds for a key press. If the ESC key is pressed, the loop breaks and the program exits. If the 'n' key is pressed, it prints a message indicating that the input is ignored.
- Hough Line Probabilistic Transform Function: The HoughLinePTransform() function is defined to encapsulate the Probabilistic Hough Line Transform operation. This function performs the transform on the dst image (containing the edges), draws the detected lines on the cdstP and src (original frame) images using the line function, and displays the result in a window named "Prob HL Transform".
- Window Destruction: After the continuous loop exits, the "video_display" window is

closed using the destroyWindow function.

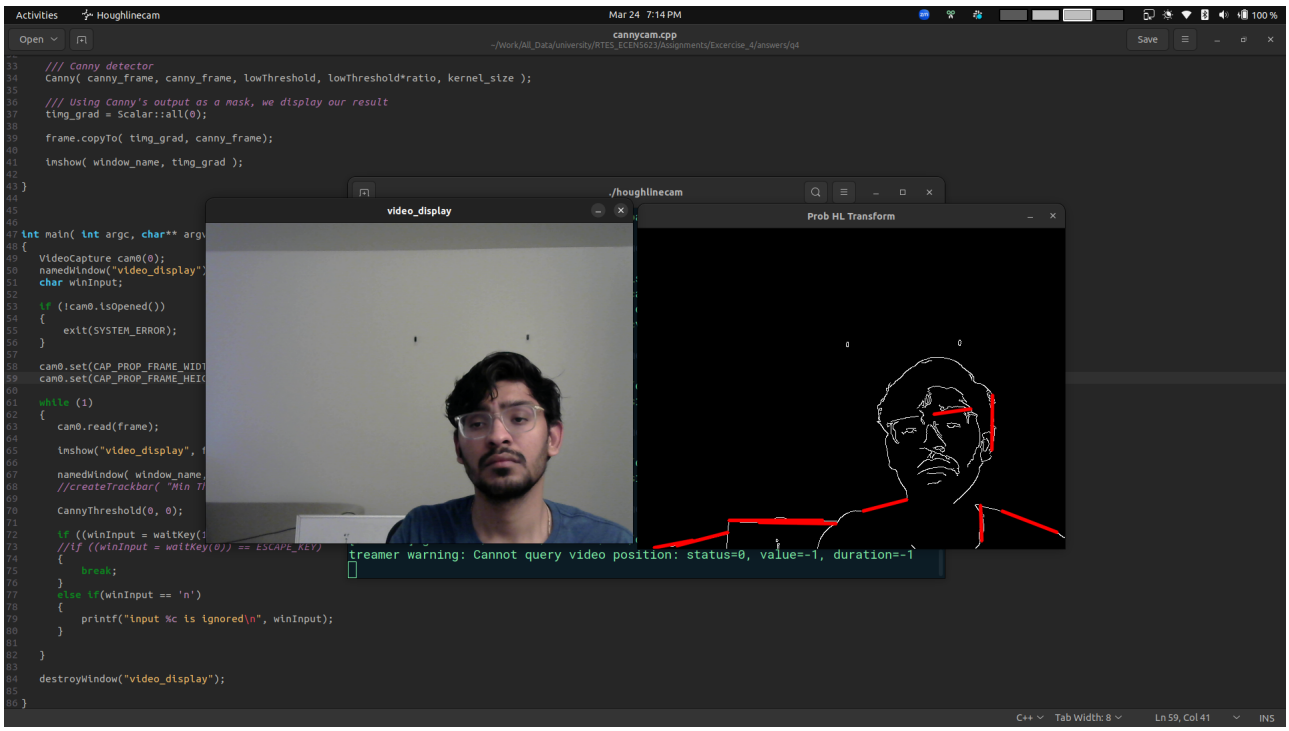


Figure 12: Hough Line

Hough Line Transform top

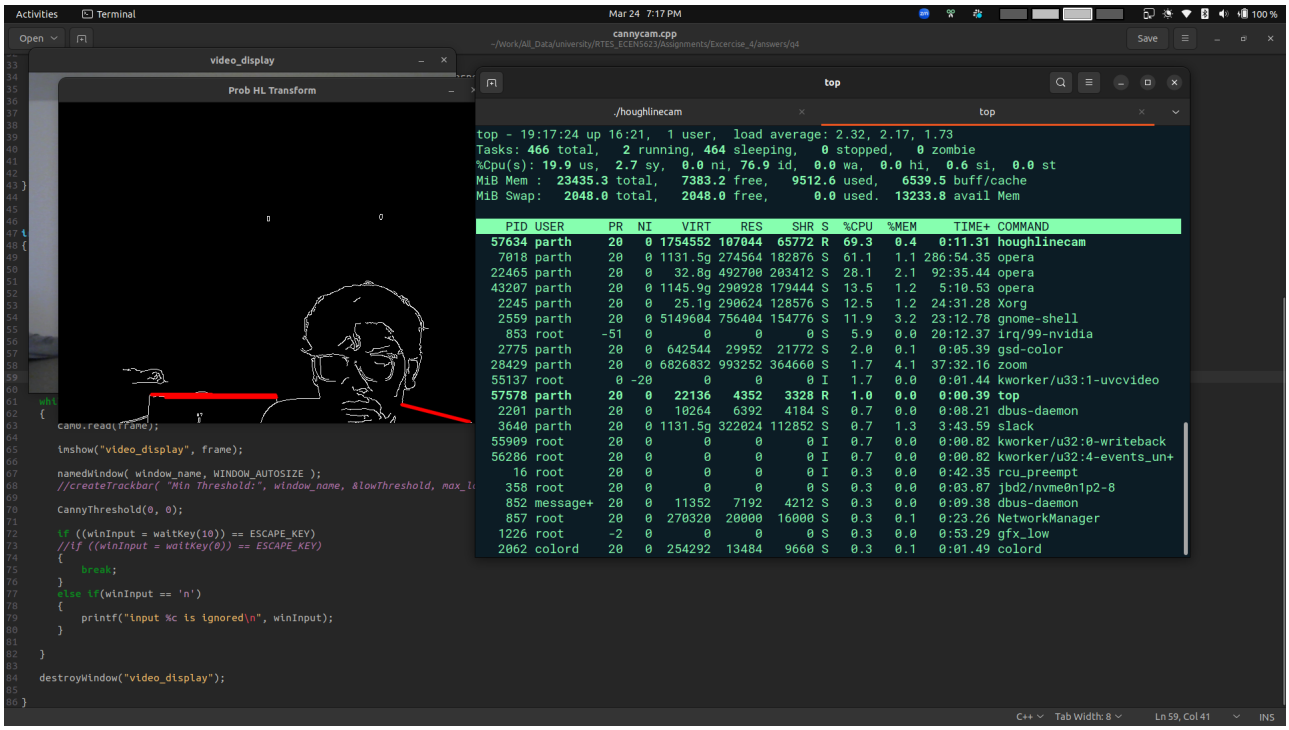


Figure 13: Hough Line Top

Hough Circle Transform Here's an explanation of the provided code

- Image Loading: This code (main() function) loads an image specified by the user or uses a default image ("../data/smarties.png") if no argument is provided.
- Image Processing:

- The loaded image (src) is directly converted to grayscale using cvtColor() and then median blur is applied to the grayscale image.
- Circle Detection: Detect circles in the processed grayscale image using the Hough Circle Transform (HoughCircles() function).
- Circle Drawing: After detecting circles, draw circles on the original color image (src) using the circle() function. Image Display: Finally, display the image with detected circles using imshow() and wait for a key press using waitKey() before exiting.

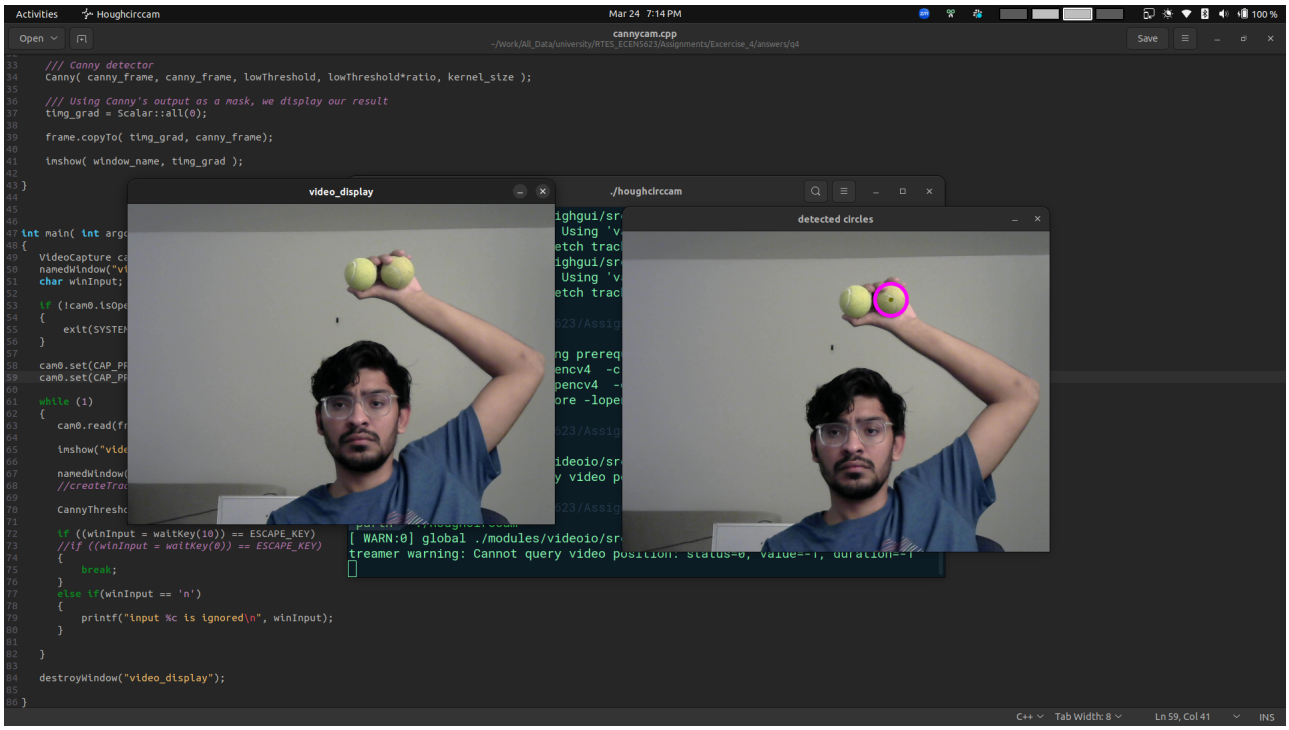


Figure 14: Hough Circle

Hough Circle Transform top

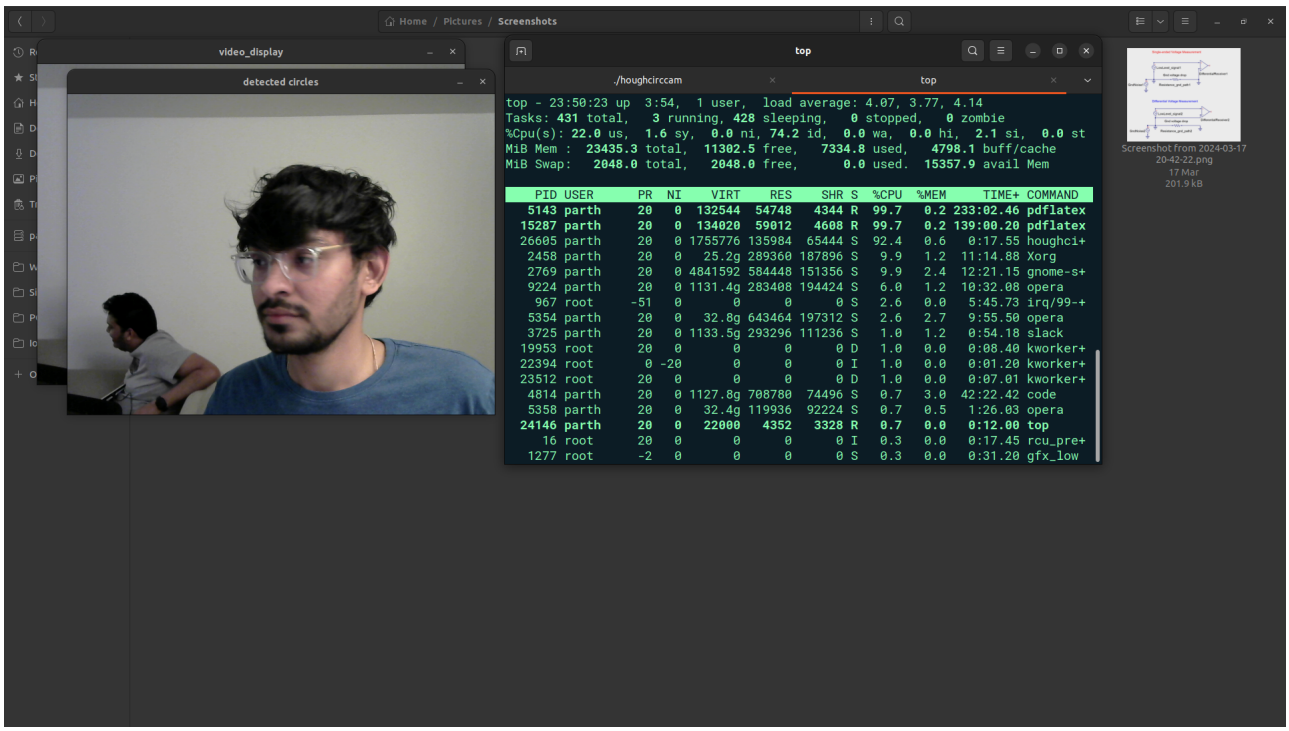


Figure 15: Hough Circle top

5 Question 5

Q: Using a Logitech C2xx, 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 (there should be a logging thread) to analyze the potential throughput. You should then get at least 9 separate datasets running 1 transformation at a time. Based on average analysis, pick a reasonable soft real-time deadline $e.g. if average frame rate is 12Hz, choose a deadline of 100 milliseconds to provide some margin$ **and convert the processing to SCHED_FIFO and determine if you can meet deadlines with predictability and measure statistical jitter in the frame rate relative to your deadline for both the default scheduler and SCHED_FIFO in each case.**

Answer:

(a) Design Concepts (Flow Chart or Diagram)

- Flow Chart/Diagram Creation:

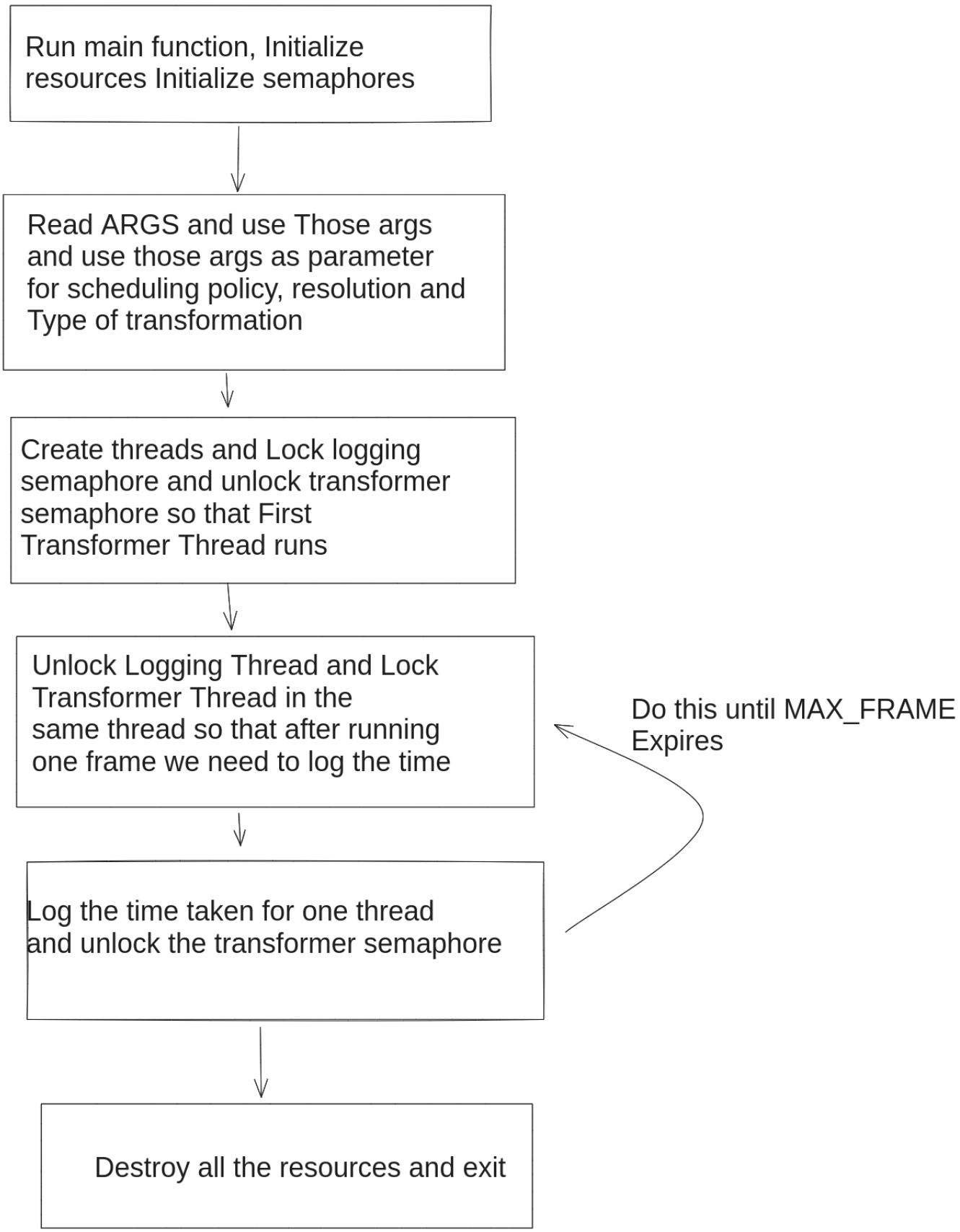


Figure 16: Flowchart

- Components to Include:
  - Video Capture and Processing: Utilizes OpenCV to capture video frames from a camera, apply image processing transformations (Canny edge detection, Hough Line and Circle

Transforms), and display the transformed images.

- Parallel Processing: Employs pthreads to create separate threads for image transformation and logging, allowing simultaneous processing and logging.
- Real-time Scheduling: Implements real-time scheduling policies (SCHED_FIFO, Sched opther) to prioritize the processing and logging tasks, aiming to meet soft real-time constraints.
- Synchronization: Uses semaphores to coordinate the execution between the transformation and logging threads, ensuring orderly processing and data logging.
- Logging and Performance Analysis: Records performance metrics (execution time, frame rate, scheduling policy, resolution, transformation type) to a CSV file for further analysis.

(b) Algorithm Analysis

- Transformation Descriptions: Describe each of the three chosen transformations in detail. For example, you might select grayscale conversion, edge detection, and Gaussian blur as your transformations. Explain the purpose of each transformation and its expected impact on processing time.
- Scheduling Approach:
  - Semaphore Initialization
    Two semaphores, transformation_semaphore and logging_semaphore, are initialized at the beginning of the program. These semaphores are used to synchronize the transformation of video frames (using different image processing techniques like Canny, Hough Line, or Hough Circle Transform) and the logging of performance metrics.
  - Synchronization Between Processing and Logging
    Transformation Semaphore (transformation_semaphore): This semaphore is used to control the execution flow between capturing/processing video frames and the logging thread. Initially, the semaphore is posted (incremented) to allow the transformation thread to start processing a frame.
    Logging Semaphore (logging_semaphore): Conversely, this semaphore is waited on (decremented) by the logging thread to ensure it logs performance data only after a frame has been processed.
    Execution Flow Video Frame Processing: The transformation thread starts by capturing a frame from the video feed. After processing the frame (applying the selected image transformation technique), the thread posts (increments) the logging_semaphore, signaling that the logging thread can now proceed to log the performance data for that frame.

    Performance Logging: The logging thread, upon being allowed to proceed (after the logging_semaphore is posted by the transformation thread), calculates the processing time and logs the performance metrics. Once logging is complete, it posts (increments) the transformation_semaphore, indicating that the transformation thread can process the next frame.

    Synchronization Loop: This process of alternately waiting on and posting semaphores creates a synchronized loop between the transformation and logging threads, ensuring that performance data for each frame is logged sequentially and accurately reflects the processing times.

    Soft Exit Mechanism The soft_exit flag is used as a soft mechanism to exit the loops in both transformation and logging threads safely. It ensures that both threads can complete their current iteration before closing, maintaining data integrity and avoiding premature termination that could lead to missed logging or incomplete frame processing.

(c) Output Screenshots

- Implementation Details: Briefly describe how the prototype was implemented, including the programming language, libraries (e.g., OpenCV for transformations), and tools used.
- Data Collection: Shows the resolution, scheduler policy, and the number of processors. The scheduler policy changes from SCHED_OTHER to SCHED_FIFO, indicating a switch to a real-time scheduling policy for more predictable execution.
- Execution Logs:
  Each "working log" entry shows the frame processing time and calculates the frame rate as the inverse of this time.
  Variability in execution time (jitter) is observable through the differences in "total_time taken for 1 frame" across frames.

- The Average FPS is calculated over the run, giving a measure of overall performance. The average execution time gives an idea of how long, on average, each frame takes to process.
- Total deadline miss: Indicates how many times the frame processing failed to meet the soft deadline, reflecting on the real-time performance and system load. For –transform=Canny

```
CSV header is correct.
Resolution: 640x480
Scheduler Policy: 1
This system has 12 processors configured and 12 processors available.
Before adjustments to scheduling policy:
Pthread Policy is SCHED_OTHER
After adjustments to scheduling policy:
Pthread Policy is SCHED_FIFO
Setting thread 0 to core 0
Setting thread 1 to core 0
Threads created
Joining thread 0
working log[ WARN:0] global ./modules/videoio/src/cap_gstreamer.cpp (1100)
 open OpenCV | GStreamer warning: Cannot query video position: status=0,
 value=-1, duration=-1
working logcurrent fps : 6.800981 , total_time taken for 1 frame: 0.147038
working logcurrent fps : 7.669834 , total_time taken for 1 frame: 0.130381
working logcurrent fps : 7.822449 , total_time taken for 1 frame: 0.127837
working logcurrent fps : 7.325486 , total_time taken for 1 frame: 0.136510
working logcurrent fps : 7.348487 , total_time taken for 1 frame: 0.136082
working logcurrent fps : 7.439587 , total_time taken for 1 frame: 0.134416
working logcurrent fps : 7.400848 , total_time taken for 1 frame: 0.135120
working logcurrent fps : 7.616983 , total_time taken for 1 frame: 0.131286
working logcurrent fps : 7.260292 , total_time taken for 1 frame: 0.137736
Thread 0 joined successfully.
Joining thread 1
Thread 1 joined successfully.
Average FPS 8.220948, Average time 0.121640
Total deadline miss : 5
```

For -transform=HoughLine

```
CSV header is correct.
Resolution: 640x480
Scheduler Policy: 1
This system has 12 processors configured and 12 processors available.
Before adjustments to scheduling policy:
Pthread Policy is SCHED_OTHER
After adjustments to scheduling policy:
Pthread Policy is SCHED_FIFO
Setting thread 0 to core 0
Setting thread 1 to core 0
WoringThreads created
Joining thread 0
working log[ WARN:0] global ./modules/videoio/src/cap_gstreamer.cpp (1100)
open OpenCV | GStreamer warning: Cannot query video position: status=0,
value=-1, duration=-1
working logcurrent fps : 5.931778 , total_time taken for 1 frame: 0.168584
```

```
working logcurrent fps : 9.412008 , total_time taken for 1 frame: 0.106247
working logcurrent fps : 7.601903 , total_time taken for 1 frame: 0.131546
working logcurrent fps : 7.402297 , total_time taken for 1 frame: 0.135093
working logcurrent fps : 7.084185 , total_time taken for 1 frame: 0.141160
working logcurrent fps : 7.610951 , total_time taken for 1 frame: 0.131390
working logcurrent fps : 7.393532 , total_time taken for 1 frame: 0.135253
working logcurrent fps : 7.787055 , total_time taken for 1 frame: 0.128418
working logcurrent fps : 7.366140 , total_time taken for 1 frame: 0.135756
Thread 0 joined successfully.
Joining thread 1
Thread 1 joined successfully.
Average FPS 8.240986, Average time 0.121345
Total deadline miss : 5


```

Fpr -transform=HoughCircle

```
CSV header is correct.
Resolution: 640x480
Scheduler Policy: 1
This system has 12 processors configured and 12 processors available.
Before adjustments to scheduling policy:
Pthread Policy is SCHED_OTHER
After adjustments to scheduling policy:
Pthread Policy is SCHED_FIFO
Setting thread 0 to core 0
Setting thread 1 to core 0
Threads created
Joining thread 0
working log[ WARN:0] global ./modules/videoio/src/cap_gstreamer.cpp (1100)
open OpenCV | GStreamer warning: Cannot query video position: status=0,
 value=-1, duration=-1
working logcurrent fps : 7.336637 , total_time taken for 1 frame: 0.136302
working logcurrent fps : 6.928957 , total_time taken for 1 frame: 0.144322
working logcurrent fps : 8.009877 , total_time taken for 1 frame: 0.124846
working logcurrent fps : 7.141519 , total_time taken for 1 frame: 0.140026
working logcurrent fps : 7.693228 , total_time taken for 1 frame: 0.129984
working logcurrent fps : 7.618371 , total_time taken for 1 frame: 0.131262
working logcurrent fps : 7.320480 , total_time taken for 1 frame: 0.136603
working logcurrent fps : 7.581601 , total_time taken for 1 frame: 0.131898
working logcurrent fps : 7.249677 , total_time taken for 1 frame: 0.137937
Thread 0 joined successfully.
Joining thread 1
Thread 1 joined successfully.
Average FPS 8.242794, Average time 0.121318
Total deadline miss : 5

```

- Bash Script: This bash script is designed to automate the testing of a program (./program) with different sets of parameters: transformation types, resolutions, and scheduler policies. It systematically iterates through each combination of these parameters, executes the program with them, and checks for any errors during execution.
- Python Script: This Python script is designed to load data from a CSV file, process and analyze this data, and finally plot specific portions of the data, focusing on execution times. It uses the pandas library for data manipulation and matplotlib for plotting

(d) Final Predictable Response Jitter Analysis

- Jitter Analysis Methodology: Describe how jitter was calculated, considering the variation in frame processing times relative to the set deadline. Explain the statistical methods used to analyze jitter.
- Graphical Representation:
  - Graph 1 (Jitter vs. Number of Frames):

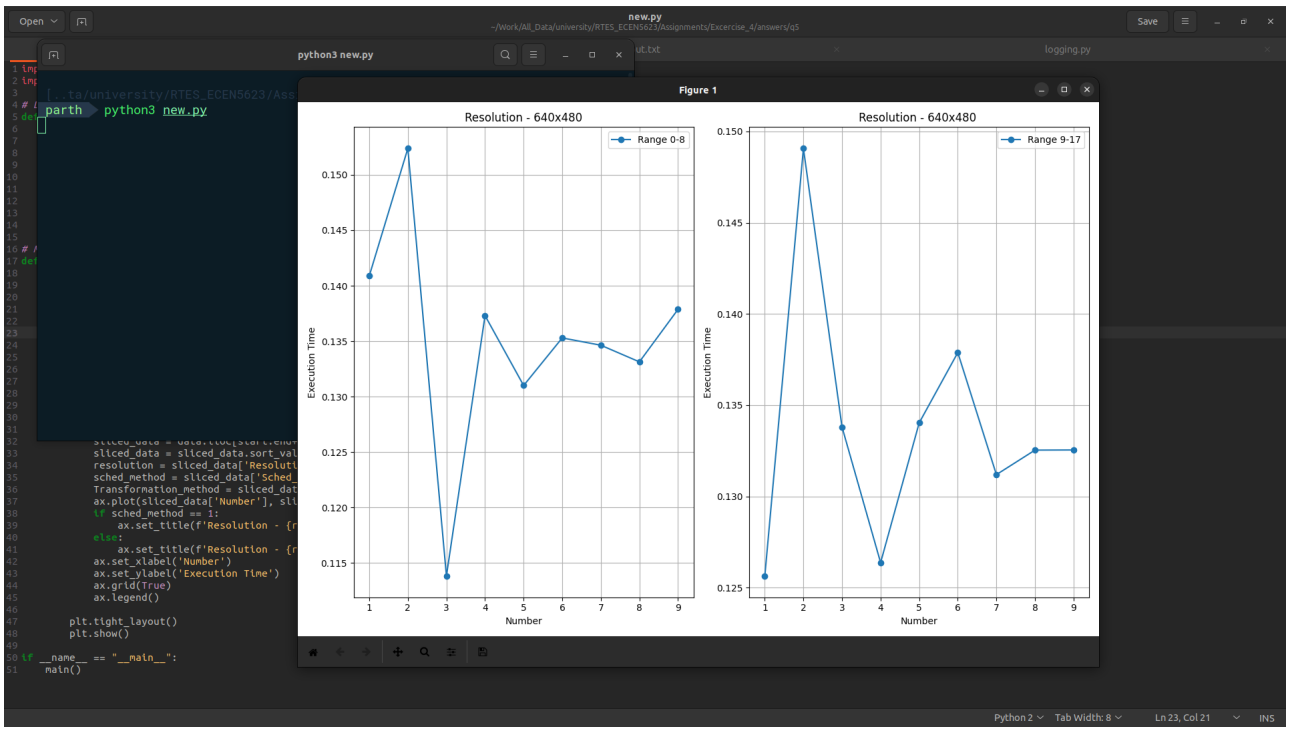


Figure 17: Graph 1

– Graph 2 (Number of Frames & Execution Time):

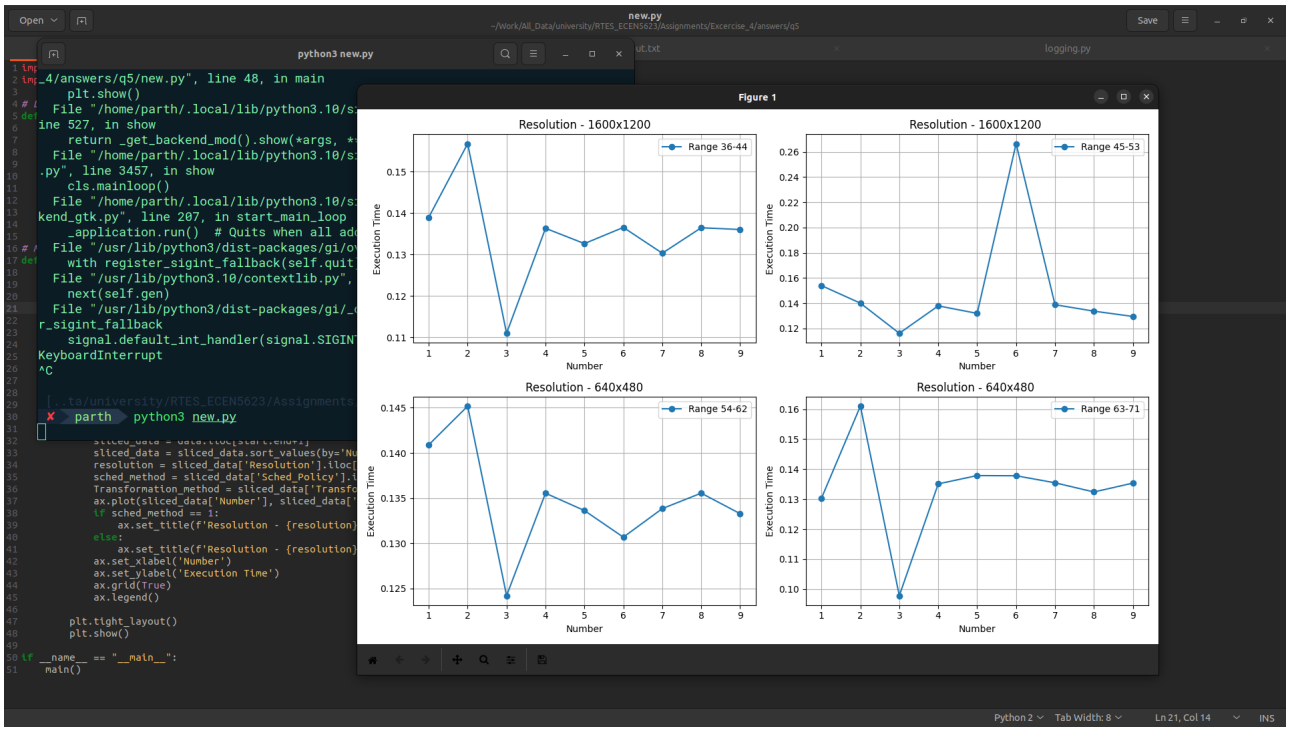


Figure 18: Graph 2

– Graph 2 (Number of Frames & Execution Time):

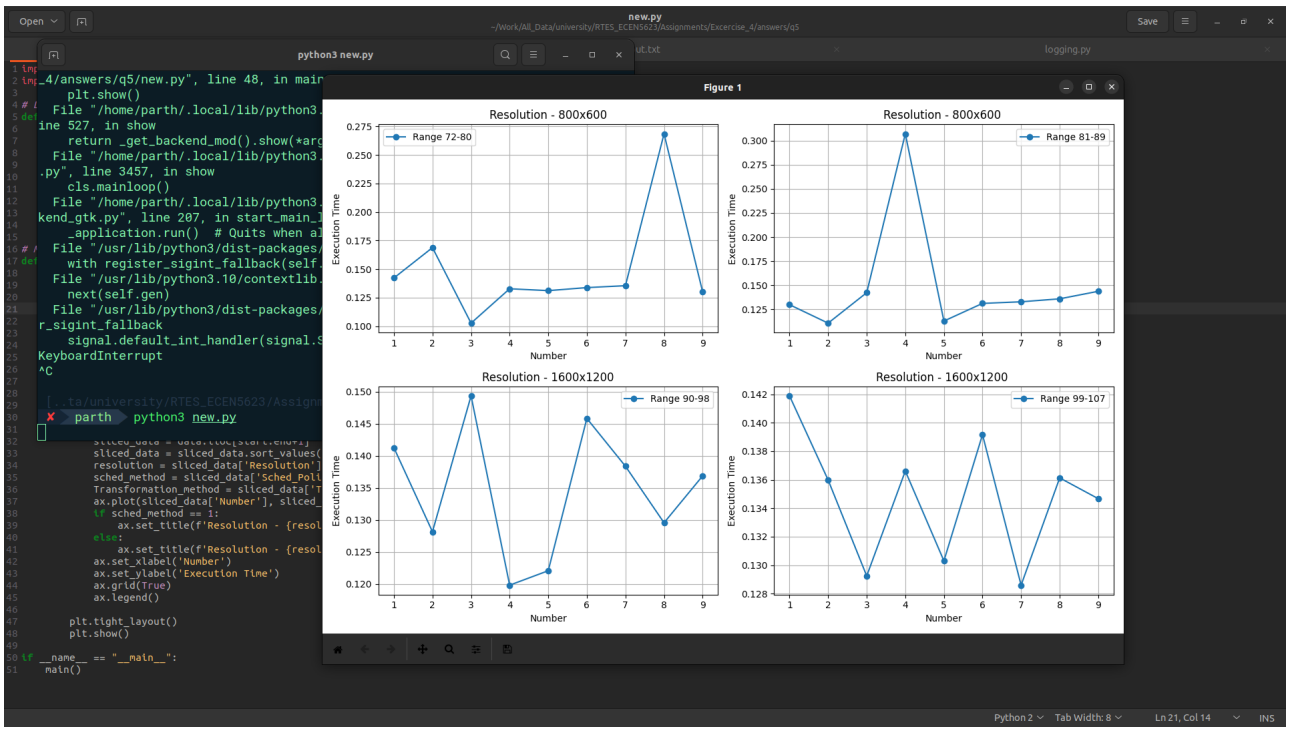


Figure 19: Graph 3

– Graph 2 (Number of Frames & Execution Time):

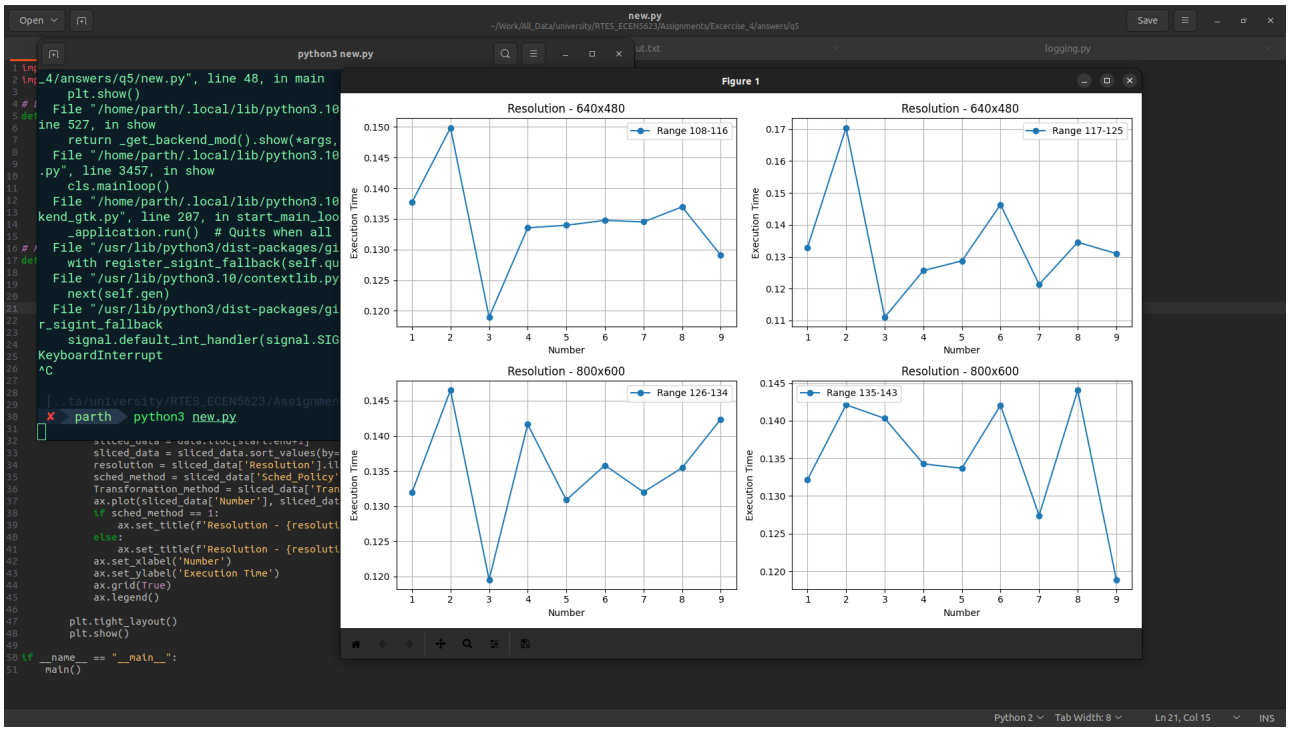


Figure 20: Graph 4

6 Question 6

Q: [10 points] Demonstrate the results and answer questions from the TA regarding the code you developed in #2, #3, #4, and #5.

7 References

1. ECEN 5623 Lecture slides material and example codes.
2. REAL-TIME EMBEDDED COMPONENTS AND SYSTEMS with LINUX and RTOS, Sam Siewert John Pratt (Chapter 6, 7 & 8).
3. Exercise 4 requirements included links and documentation.

Appendices

A C/Cpp/Python Code for the Implementation

Ex_3_script.sh

```
#!/bin/bash

matrix=("Canny" "HoughLine" "HoughCircle")
resolutions=("640x480" "800x600" "1600x1200")
sched_policies=("SCHED_FIFO" "SCHED_OTHER")

rm example.csv
touch example.csv

# Iterate through each combination of parameters
for mat in "${matrix[@]}"; do
    for resolution in "${resolutions[@]}"; do
        for policy in "${sched_policies[@]}"; do
            echo "Running: ./program --transformation=${mat} --resolution=${resolution} --sched_policy=${policy}"
            sudo ./program --transformation=$mat --resolution=$resolution --sched_policy=$policy
            return_value="$?"
            # Check if the program exited with an error
            if [ "$return_value" -ne 0 ]; then
                echo "Error running program with parameters --transformation=${mat} --resolution=${resolution} --sched_policy=${policy}, exiting."
                exit -1
            fi
        done
    done
done

```

example.py

```python
import matplotlib.pyplot as plt
import pandas as pd

# Load data from CSV file
def load_data_from_csv(filename):
    try:
        df = pd.read_csv(filename)
        return df
    except FileNotFoundError:
        print(f"File {filename} not found.")
        return None
    except Exception as e:
        print(f"An error occurred while loading data: {e}")
        return None

# Plot data
def plot_data(dataframe):
    if dataframe is not None:
        x = dataframe.iloc[:, 0]  # Assuming first column is x-axis data
        y = dataframe.iloc[:, 1]  # Assuming second column is y-axis data

        plt.plot(x, y)
        plt.xlabel('X Label')
        plt.ylabel('Y Label')
        plt.title('Data Plot')
        plt.grid(True)
        plt.show()

# Main function
def main():
    csv_filename = 'example.csv'
    data = load_data_from_csv(csv_filename)

    # Grouping the data by 'Sched_Policy' and 'Transform', then calculate standard deviation
    grouped_std_dev = data.groupby(['Sched_Policy', 'Transform', 'Resolution'])['Execution_time'].std().reset_index()
    print(grouped_std_dev);

    grouped_std_dev


    # if data is not None:
    #     plot_data(data)

if __name__ == "__main__":
    main()

```

program.cpp

```cpp
#include "opencv2/imgcodecs.hpp"
#include "opencv2/highgui.hpp"
#include "opencv2/imgproc.hpp"
#include <iostream>
#include <string>
#include <map>
#include <cstdlib> // For std::exit and EXIT_FAILURE
#include <pthread.h>
#include <sched.h>
#include <time.h>
#include <fstream>

#include <pthread.h>
#include <stdio.h>
#include <time.h>
#include <aio.h>
#include <math.h>
#include <unistd.h>
#include <errno.h>
#include <stdlib.h>
#include <sched.h>
#include <stddef.h>
#include <sys/sysinfo.h>
#include <semaphore.h>

#define MAX_FRAME 10
#define NUM_OF_THREAD 2
#define ESCAPE_KEY (27)
#define SYSTEM_ERROR (-1)
#define TARGET_CORE 0

#define SOFT_DEADLINE 0.135 // ms

sem_t transformation_semaphore, logging_semaphore;
static int total_deadline_miss = 0;
bool initial = true;

void *(*transformationType)(void *);
std::string resolution;
int schedPolicy = SCHED_FIFO; // Default policy
std::string transformationStr;
volatile int soft_exit = 0;
double sum_of_execution = 0;

std::string filePath = "./example.csv";

struct timespec update_interval;
struct timespec last_update_interval;
struct timespec start_interval;
struct timespec end_interval;

namespace canny
{
```

```
    int lowThreshold = 0;
    const int max_lowThreshold = 100;
    const int ratio = 3;
    const int kernel_size = 3;
    const char *window_name = "Edge Map";
}

typedef struct
{
    int threadId;
    int width;
    int height;
    cv::Mat frame;
    cv::Mat intermediate_src;
    cv::Mat output;
    std::string window_name;

} ThreadArgs_t;

typedef struct
{
    int period;
    int burst_time;
    struct sched_param priority_param;
    void *(*thread_handle)(void *);
    pthread_t thread;
    ThreadArgs_t thread_args;
    void *return_Value;
    pthread_attr_t attribute;
    int target_cpu;
} RmTask_t;

// Mat frame;
void print_scheduler(void)
{
    int schedType;
    schedType = sched_getscheduler(getpid());
    switch (schedType)
    {
    case SCHED_FIFO:
        printf("Pthread Policy is SCHED_FIFO\n");
        break;
    case SCHED_OTHER:
        printf("Pthread Policy is SCHED_OTHER\n");
        break;
    case SCHED_RR:
        printf("Pthread Policy is SCHED_OTHER\n");
        break;
    default:
        printf("Pthread Policy is UNKNOWN\n");
    }
}

void appendDataToCsv(int max_frame, float total_time)
{
    // Open the file in append mode
```

```
110     std::ofstream file(filePath, std::ios::app);
111 
112     if (!file.is_open())
113     {
114         std::cerr << "Failed to open the file for appending." << std::endl;
115         return;
116     }
117 
118     // Write the new data to the file
119     file << MAX_FRAME - max_frame << "," << total_time << "," << (1 / total_time) << "," << schedPolicy << "," << resolution << "," << transformationStr << std::endl;
120 
121     // Close the file
122     file.close();
123 }
124 
125 void HoughLinePTransform(RmTask_t *data)
126 {
127 
128     std::vector<cv::Vec4i> linesP; // Will hold the results of the detection
129     cv::HoughLinesP(data->thread_args.intermediate_src, linesP, 1, CV_PI / 180, 50, 50, 10); // Runs the actual detection
130 
131     // Draw the lines
132     for (size_t i = 0; i < linesP.size(); i++)
133     {
134         cv::Vec4i l = linesP[i];
135         cv::line(data->thread_args.output, cv::Point(l[0], l[1]), cv::Point(l[2], l[3]), cv::Scalar(0, 0, 255), 3, cv::LINE_AA);
136         cv::line(data->thread_args.frame, cv::Point(l[0], l[1]), cv::Point(l[2], l[3]), cv::Scalar(0, 0, 255), 3, cv::LINE_AA);
137     }
138     cv::imshow(data->thread_args.window_name, data->thread_args.output);
139 }
140 
141 // Function to detect and draw circles in an image using the Hough Transform
142 void HoughCircleCam(RmTask_t *data)
143 {
144 
145     cvtColor(data->thread_args.frame, data->thread_args.intermediate_src, cv::COLOR_BGR2GRAY);
146     cv::medianBlur(data->thread_args.intermediate_src, data->thread_args.intermediate_src, 5);
147     std::vector<cv::Vec3f> circles;
148 
149     cv::HoughCircles(data->thread_args.intermediate_src, circles, cv::HOUGH_GRADIENT, 1,
150                      data->thread_args.intermediate_src.rows / 16, // Change this value to detect circles with different distances to each other
151                      100, 30, 1, 30                                // Change the last two parameters (min_radius & max_radius) to detect larger circles
152     );
153 
154     for (size_t i = 0; i < circles.size(); i++)
155     {
156         cv::Vec3i c = circles[i];
157         cv::Point center = cv::Point(c[0], c[1]); // Circle center
```

```
158         circle(data->thread_args.frame, center, 1, cv::Scalar(0, 100, 100), 3, cv::LINE_AA);
159         int radius = c[2];
160         circle(data->thread_args.frame, center, radius, cv::Scalar(255, 0, 255), 3, cv::LINE_AA);
161     }
162
163     cv::imshow("detected circles", data->thread_args.frame);
164 }
165
166 void CannyThreshold(RmTask_t *data)
167 {
168     cvtColor(data->thread_args.frame, data->thread_args.intermediate_src, cv::COLOR_BGR2GRAY);
169
170     /// Reduce noise with a kernel 3x3
171     cv::blur(data->thread_args.intermediate_src, data->thread_args.output, cv::Size(3, 3));
172
173     /// Canny detector
174     cv::Canny(data->thread_args.output, data->thread_args.output, canny::lowThreshold, canny::lowThreshold * canny::ratio, canny::kernel_size);
175
176     /// Using Canny's output as a mask, we display our result
177     cv::Mat timg_grad;
178
179     // Initialize `timg_grad` to be the same size and type as the source frame, but filled with zeros
180     // Assuming `data->thread_args.frame` is a cv::Mat
181     timg_grad = cv::Mat::zeros(data->thread_args.frame.size(), data->thread_args.frame.type());
182
183     // Using Canny's output as a mask, copy the frame to `timg_grad` wherever the mask is non-zero
184     data->thread_args.frame.copyTo(timg_grad, data->thread_args.output);
185
186     // Display the result
187     cv::imshow(data->thread_args.window_name, timg_grad);
188 }
189
190 void *HoughLineTransform(void *args)
191 {
192     printf("Woring");
193     RmTask_t *data = static_cast<RmTask_t *>(args);
194     data->thread_args.window_name = "Houghline Transformation";
195     cv::VideoCapture cam0(0);
196     cv::namedWindow("video_display");
197     char winInput;
198
199     if (!cam0.isOpened())
200     {
201         exit(SYSTEM_ERROR);
202     }
203
204     cam0.set(cv::CAP_PROP_FRAME_WIDTH, data->thread_args.width);
205     cam0.set(cv::CAP_PROP_FRAME_HEIGHT, data->thread_args.height);
206
207     int max_frame = MAX_FRAME;
```

```
208     clock_gettime(CLOCK_REALTIME, &start_interval);
209 
210     while (max_frame)
211     {
212         sem_wait(&transformation_semaphore);
213         cam0.read(data->thread_args.frame);
214         cv::imshow("video_display", data->thread_args.frame);
215         cv::Canny(data->thread_args.frame, data->thread_args.intermediate_src, 80,
    240, 3);
216         cv::cvtColor(data->thread_args.intermediate_src, data->thread_args.output,
    cv::COLOR_GRAY2BGR);
217         cv::namedWindow(data->thread_args.window_name, cv::WINDOW_AUTOSIZE);
218         HoughLinePTransform(data);
219         if ((winInput = cv::waitKey(10)) == ESCAPE_KEY)
220         {
221             soft_exit = 1;
222             sem_post(&logging_semaphore);
223             cv::destroyAllWindows();
224             break;
225         }
226         max_frame--;
227         sem_post(&logging_semaphore);
228     }
229 
230     return nullptr;
231 }
232 
233 void *HoughCircleTransform(void *args)
234 {
235 
236     RmTask_t *data = static_cast<RmTask_t *>(args);
237     data->thread_args.window_name = "HoughCircle Transformation";
238     cv::VideoCapture cam0(0);
239     cv::namedWindow("video_display");
240     char winInput;
241     if (!cam0.isOpened())
242     {
243         exit(SYSTEM_ERROR);
244     }
245     cam0.set(cv::CAP_PROP_FRAME_WIDTH, data->thread_args.width);
246     cam0.set(cv::CAP_PROP_FRAME_HEIGHT, data->thread_args.height);
247 
248     int max_frame = MAX_FRAME;
249     clock_gettime(CLOCK_REALTIME, &start_interval);
250     while (max_frame)
251     {
252         sem_wait(&transformation_semaphore);
253         cam0.read(data->thread_args.frame);
254         cv::imshow("video_display", data->thread_args.frame);
255         HoughCircleCam(data);
256         if ((winInput = cv::waitKey(10)) == ESCAPE_KEY)
257         {
258             soft_exit = 1;
259             sem_post(&logging_semaphore);
260             cv::destroyAllWindows();
261             break;
262         }
```

```
263         else if (winInput == 'n')
264         {
265             printf("input %c is ignored\n", winInput);
266         }
267         max_frame--;
268         sem_post(&logging_semaphore);
269     }
270 
271     return nullptr;
272 }
273 
274 void *CannyTransform(void *args)
275 {
276 
277     RmTask_t *data = static_cast<RmTask_t *>(args);
278     data->thread_args.window_name = "Canny Transformation";
279     cv::VideoCapture cam0(0);
280     cv::namedWindow("video_display");
281     char winInput;
282 
283     if (!cam0.isOpened())
284     {
285         exit(SYSTEM_ERROR);
286     }
287 
288     cam0.set(cv::CAP_PROP_FRAME_WIDTH, data->thread_args.width);
289     cam0.set(cv::CAP_PROP_FRAME_HEIGHT, data->thread_args.height);
290 
291     int max_frame = MAX_FRAME;
292 
293     clock_gettime(CLOCK_REALTIME, &start_interval);
294     while (max_frame)
295     {
296         sem_wait(&transformation_semaphore);
297 
298         cam0.read(data->thread_args.frame);
299 
300         cv::imshow("Canny Video", data->thread_args.frame);
301 
302         cv::namedWindow(data->thread_args.window_name, cv::WINDOW_AUTOSIZE);
303 
304         CannyThreshold(data);
305 
306         if ((winInput = cv::waitKey(10)) == ESCAPE_KEY)
307         {
308             soft_exit = 1;
309             sem_post(&logging_semaphore);
310             cv::destroyAllWindows();
311             break;
312         }
313 
314         max_frame--;
315         sem_post(&logging_semaphore);
316     }
317 
318     return nullptr;
```

```
319 }
320
321 void *LoggingThread(void *args)
322 {
323     int max_frame = MAX_FRAME;
324     clock_gettime(CLOCK_REALTIME, &last_update_interval);
325     while (max_frame)
326     {
327         printf("working log");
328
329         sem_wait(&logging_semaphore);
330
331         clock_gettime(CLOCK_REALTIME, &update_interval);
332         float total_sec = update_interval.tv_sec - last_update_interval.tv_sec;
333         float total_ns = ((float)(update_interval.tv_nsec -
    last_update_interval.tv_nsec) / 1000000000);
334         float total_time = total_sec + total_ns;
335
336         if (!initial)
337         {
338
339             if (total_time > SOFT_DEADLINE)
340             {
341                 total_deadline_miss++;
342             }
343             sum_of_execution += total_time;
344
345             printf("current fps/jitter: update_interval - last_update_interval : %f ,
    total_time taken for 1 frame: %f \n\r", (1 / total_time), total_time);
346
347             appendDataToCsv(max_frame, total_time);
348
349         }
350
351         if (soft_exit)
352         {
353             printf("Exiting thread 2\n\r");
354             break;
355         }
356         sem_post(&transformation_semaphore);
357         last_update_interval = update_interval;
358         max_frame--;
359         initial = false;
360     }
361     return nullptr;
362 }
363
364 int main(int argc, char **argv)
365 {
366
367     sem_init(&transformation_semaphore, false, 0);
368     sem_init(&logging_semaphore, false, 0);
369
370     sem_post(&transformation_semaphore);
371
372     const std::string expectedHeader = "Number,Execution_time,Frame_rate,
    Sched_Policy,Resolution,Transform";
```

```
373     std::ifstream file(filePath);
374     std::string currentHeader;
375 
376     if (file.is_open())
377     {
378         // Get the first line from the file
379         std::getline(file, currentHeader);
380         file.close();
381 
382         // Check if the current header matches the expected header
383         if (currentHeader != expectedHeader)
384         {
385             // The headers do not match, so we need to write the correct header
386 
387             // Store the rest of the file content
388             std::string restOfFileContent;
389             std::string line;
390             while (std::getline(file, line))
391             {
392                 restOfFileContent += line + "\n";
393             }
394 
395             // Write the correct header and the rest of the file
396             std::ofstream outFile(filePath);
397             outFile << expectedHeader << "\n"
398                     << restOfFileContent;
399             outFile.close();
400 
401             std::cout << "Header was corrected in the CSV file." << std::endl;
402         }
403         else
404         {
405 
406             std::cout << "CSV header is correct." << std::endl;
407         }
408     }
409     else
410     {
411         std::cerr << "Could not open the file." << std::endl;
412         return 1;
413     }
414 
415     void *(*transformationType)(void *) = CannyTransform;
416     std::map<std::string, int> schedPolicyMap = {
417         {"SCHED_FIFO", SCHED_FIFO},
418         {"SCHED_RR", SCHED_RR},
419         {"SCHED_OTHER", SCHED_OTHER}};
420 
421     std::map<std::string, void *(*)(void *)> TransformationPolicyMap = {
422         {"Canny", CannyTransform},
423         {"HoughLine", HoughLineTransform},
424         {"HoughCircle", HoughCircleTransform}};
425 
426     for (int i = 1; i < argc; i++)
427     {
428         std::string arg = argv[i];
```

```
429         if (arg.find("--transformation=") == 0)
430         {
431             transformationStr = arg.substr(std::string("--transformation=").length())
    ;
432             if (TransformationPolicyMap.find(transformationStr) !=
    TransformationPolicyMap.end())
433             {
434                 transformationType = TransformationPolicyMap[transformationStr];
435             }
436             else
437             {
438                 std::cerr << "Unknown transformation type: " << transformationStr <<
    std::endl;
439                 std::exit(EXIT_FAILURE);
440             }
441         }
442         else if (arg.find("--resolution=") == 0)
443         {
444             resolution = arg.substr(std::string("--resolution=").length());
445         }
446         else if (arg.find("--sched_policy=") == 0)
447         {
448             std::string policyStr = arg.substr(std::string("--sched_policy=")
    .length());
449             if (schedPolicyMap.find(policyStr) != schedPolicyMap.end())
450             {
451                 schedPolicy = schedPolicyMap[policyStr];
452             }
453             else
454             {
455                 std::cerr << "Unknown scheduler policy: " << policyStr << std::endl;
456                 std::exit(EXIT_FAILURE);
457             }
458         }
459     }
460
461     std::cout << "Resolution: " << resolution << std::endl;
462     std::cout << "Scheduler Policy: " << schedPolicy << std::endl;
463
464     pthread_t threads[NUM_OF_THREAD];
465     cpu_set_t threadcpu;
466
467     CPU_SET(TARGET_CORE, &threadcpu);
468
469     RmTask_t tasks[NUM_OF_THREAD] = {
470         {.period = 20,     // ms
471          .burst_time = 10, // ms
472          .priority_param = {0},
473          .thread_handle = transformationType,
474          .thread = threads[0],
475          .thread_args = {0, 0, 0},
476          .return_Value = NULL,
477          .attribute = {0, 0},
478          .target_cpu = TARGET_CORE},
479
480         {.period = 20,     // ms
481          .burst_time = 10, // ms
```

```
482             .priority_param = {0},
483             .thread_handle = LoggingThread,
484             .thread = threads[0],
485             .thread_args = {0, 0, 0},
486             .return_Value = NULL,
487             .attribute = {0, 0},
488             .target_cpu = TARGET_CORE},
489     };
490
491     pthread_attr_t attribute_flags_for_main; // for schedular type, priority
492     struct sched_param main_priority_param;
493
494     printf("This system has %d processors configured and %d processors available.\n",
    get_nprocs_conf(), get_nprocs());
495
496     printf("Before adjustments to scheduling policy:\n");
497     print_scheduler();
498
499     int rt_max_prio = sched_get_priority_max(schedPolicy);
500     int rt_min_prio = sched_get_priority_min(schedPolicy);
501
502     main_priority_param.sched_priority = rt_max_prio;
503     for (int i = 0; i < NUM_OF_THREAD; i++)
504     {
505         tasks[i].priority_param.sched_priority = rt_max_prio;
506
507         // initialize attributes
508         pthread_attr_init(&tasks[i].attribute);
509
510         pthread_attr_setinheritsched(&tasks[i].attribute, PTHREAD_EXPLICIT_SCHED);
511         pthread_attr_setschedpolicy(&tasks[i].attribute, schedPolicy);
512         pthread_attr_setschedparam(&tasks[i].attribute, &tasks[i].priority_param);
513         pthread_attr_setaffinity_np(&tasks[i].attribute, sizeof(cpu_set_t), &
    threadcpu);
514     }
515
516     pthread_attr_init(&attribute_flags_for_main);
517
518     // pthread_attr_setinheritsched(&attribute_flags_for_main,
    PTHREAD_EXPLICIT_SCHED);
519     pthread_attr_setschedpolicy(&attribute_flags_for_main, schedPolicy);
520     pthread_attr_setaffinity_np(&attribute_flags_for_main, sizeof(cpu_set_t), &
    threadcpu);
521
522     // Main thread is already created we have to modify the priority and scheduling
    scheme
523     int status_setting_schedular = sched_setscheduler(getpid(), schedPolicy, &
    main_priority_param);
524     if (status_setting_schedular)
525     {
526         printf("ERROR; sched_setscheduler rc is %d\n", status_setting_schedular);
527         perror(NULL);
528         exit(-1);
529     }
530
531     printf("After adjustments to scheduling policy:\n");
532     print_scheduler();
```

```
533
534     int width = 0, height = 0;
535     size_t xPosition = resolution.find('x');
536     if (xPosition != std::string::npos)
537     {
538         width = std::stoi(resolution.substr(0, xPosition));
539         height = std::stoi(resolution.substr(xPosition + 1));
540     }
541
542     for (int i = 0; i < NUM_OF_THREAD; i++)
543     {
544         // Create a thread
545         // First paramter is thread which we want to create
546         // Second parameter is the flags that we want to give it to
547         // third parameter is the routine we want to give
548         // Fourth parameter is the value
549         printf("Setting thread %d to core %d\n", i, TARGET_CORE);
550
551         if (pthread_create(&tasks[i].thread, &tasks[i].attribute, tasks[i]
    .thread_handle, &tasks[i]) != 0)
552         {
553             perror("Create_Fail");
554         }
555     }
556     printf("Threads created \n\r");
557     for (int i = 0; i < NUM_OF_THREAD; i++)
558     {
559         printf("Joining thread %d\n", i);
560         int join_result = pthread_join(tasks[i].thread, &tasks[i].return_Value);
561         if (join_result != 0)
562         {
563             printf("pthread_join failed for thread %d: %s\n", i,
    strerror(join_result));
564         }
565         else
566         {
567             printf("Thread %d joined successfully.\n", i);
568         }
569     }
570
571     if (pthread_attr_destroy(&tasks[0].attribute) != 0)
572         perror("attr destroy");
573     if (pthread_attr_destroy(&tasks[1].attribute) != 0)
574         perror("attr destroy");
575
576     clock_gettime(CLOCK_REALTIME, &end_interval);
577
578     float total_sec = end_interval.tv_sec - start_interval.tv_sec;
579     float total_ns = ((float)(end_interval.tv_nsec - start_interval.tv_nsec) /
    1000000000.0);
580     float total_time = (total_sec + total_ns);
581
582     printf("Average FPS %f, Average time %f \n\r", (MAX_FRAME / sum_of_execution),
    sum_of_execution/MAX_FRAME);
583     printf("Total deadline miss : %d\n\r", total_deadline_miss);
584     sem_destroy(&transformation_semaphore);
585     sem_destroy(&logging_semaphore);
```

```
586
587     return 0;
588 }
```