Project 4 - Exploration for optimal solutions in terms of FPS and PSNR

Orion Peter Fernandes - 65047854

1. Project Description (2 pts)

The project deals with optimizing real-time Canny Edge detection system with techniques such as adjusting image size, changing input parameters, multi-threading (pthreads and OpenMP) and content of captured images. The goal of this project is to understand how does psnr works and the effects of varying parameters on the image.

2. Experimental Setup (4 pts)

- Download camera_canny_psnr.cpp, calcpsnr.h, calcpsnr.c, ground_crew.h264, tiger_face.jpg, and place original canny_util.c/h in the same folder.
- Create 3 folders on the Raspberry Pi, each for the image, video and webcam input.
- Transfer the downloaded files from Canvas to each folder created.
- Edit the camera_canny_psnr.cpp file in each folder to input the image/video/webcam input. For example -

  frame = imread("tiger_face.jpg")

- Remove appropriate comments within camera_canny_psnr.cpp. and edit out the macro value of WIDTH/HEIGHT/NFRAME to 888x900.
- Compile camera_canny_psnr.cpp to create psnr executable

  $ g++ `pkg-config --cflags --libs opencv` -I. canny_util.c calcpsnr.c camera_canny_psnr.cpp -o psnr

- Run the psnr executable created for 5 different values

  ./psnr 1.0 0.2 0.6

- In the video folder, repeat the same process as for the image with necessary edits to camera_canny_psnr.cpp to input ground_crew.h264 and edit the macro value of width/height to 1280x780. Run the psnr executable with 5 different values.
- Connect the RPi Camera and take a picture in the webcam folder

  raspistill -o screen.jpg

- Edit out the macro value of WIDTH/HEIGHT to 2592x1944 (max. resolution) in canny_camera_psnr.cpp
- Compile the edited camera_canny_psnr.cpp to create the psnr executable and run it with 5 different values.
- Note down average psnr observations for each input.
- Install imagemagick software on Linux to resize resolutions using

sudo apt-get install imagemagick

- Repeat the procedures for varying resolutions using resize().

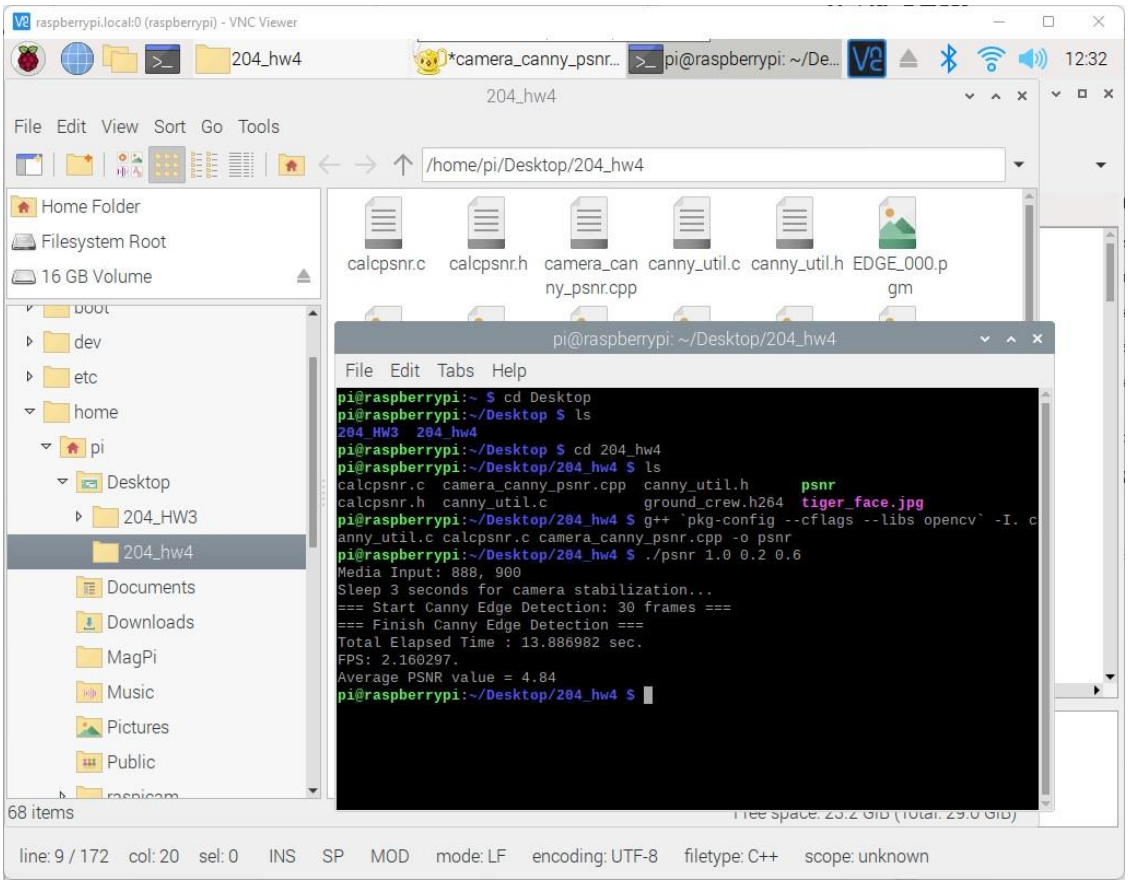
3. Results (6 pts)

- For tiger_face.jpg

Tiger_face.jpg

RAW – tiger_face.jpg

Canny Edge – tiger_face.jpg



```
pi@raspberrypi:~/Desktop/204_hw4/image $ ./psnr 0.5 0.5 0.5
Media Input: 888, 900
Sleep 3 seconds for camera stabilization...
=== Start Canny Edge Detection: 30 frames ===
=== Finish Canny Edge Detection ===
Total Elapsed Time : 15.245458 sec.
FPS: 1.967799.
Average PSNR value = 4.89
pi@raspberrypi:~/Desktop/204_hw4/image $ ./psnr 0.3 0.6 0.9
Media Input: 888, 900
Sleep 3 seconds for camera stabilization...
=== Start Canny Edge Detection: 30 frames ===
=== Finish Canny Edge Detection ===
Total Elapsed Time : 14.955252 sec.
FPS: 2.005984.
Average PSNR value = 4.78
pi@raspberrypi:~/Desktop/204_hw4/image $ ./psnr 0.2 0.4 0.8
Media Input: 888, 900
Sleep 3 seconds for camera stabilization...
=== Start Canny Edge Detection: 30 frames ===
=== Finish Canny Edge Detection ===
Total Elapsed Time : 13.243729 sec.
FPS: 2.265223.
Average PSNR value = 4.88
```

```
pi@raspberrypi:~/Desktop/204_hw4/image $ ./psnr 0.6 0.8 0.3
Media Input: 888, 900
Sleep 3 seconds for camera stabilization...
=== Start Canny Edge Detection: 30 frames ===
=== Finish Canny Edge Detection ===
Total Elapsed Time : 19.448243 sec.
FPS: 1.542556.
Average PSNR value = 4.88
```

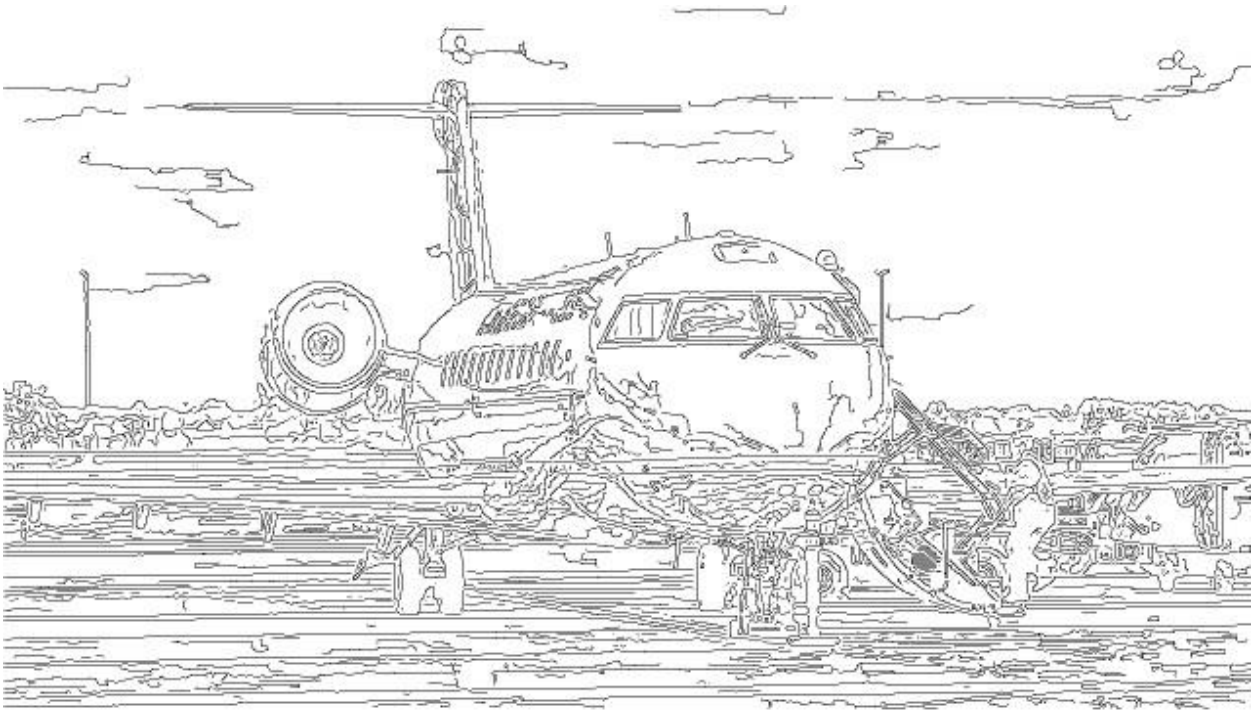
camera_canny_psnr.cpp execution for tiger_face.jpg

Observations

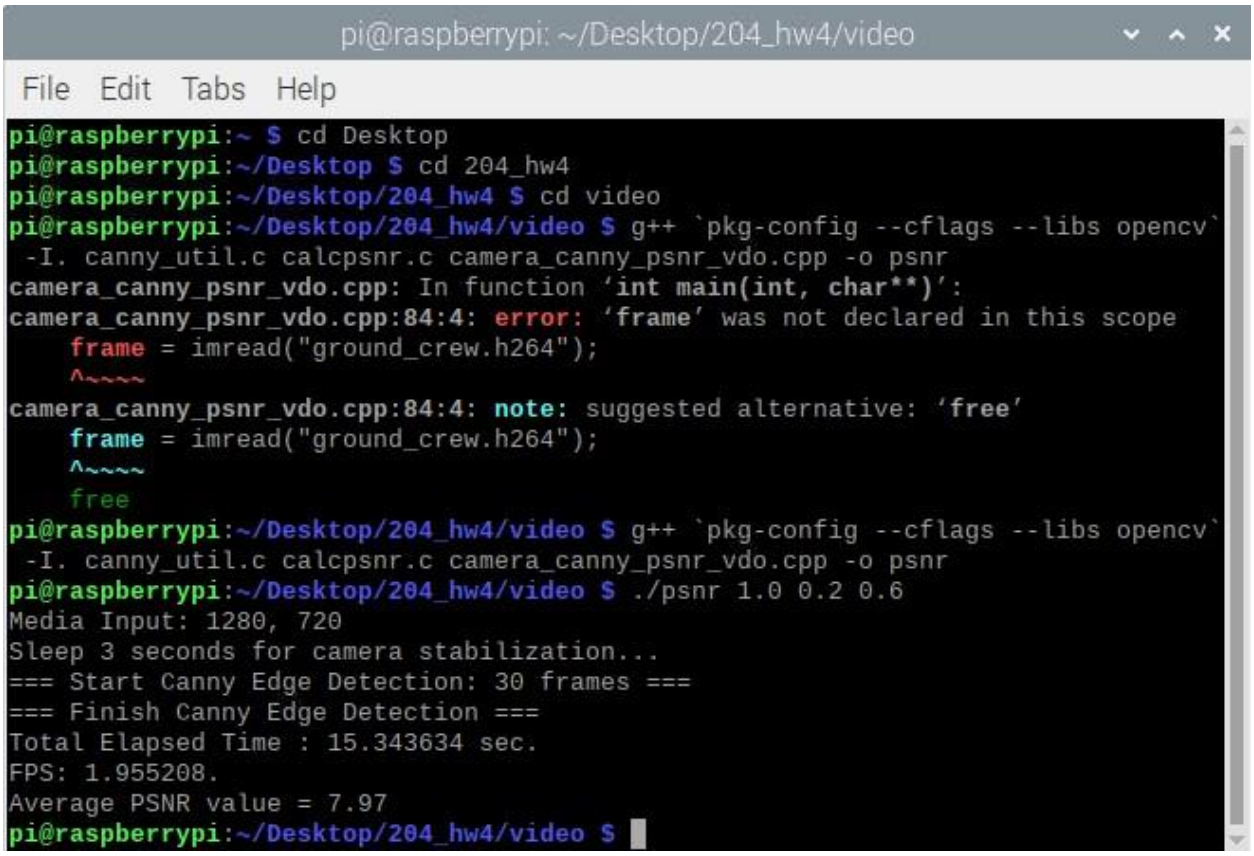
Trial	sigma	tlow	thigh	FPS	Average PSNR
1	1.0	0.2	0.6	2.160297	4.84
2	0.5	0.5	0.5	1.967799	4.89
3	0.3	0.6	0.9	2.005984	4.78
4	0.2	0.4	0.8	2.265223	4.88
5	0.6	0.8	0.3	1.542556	4.88

- For ground_crew.h264

RAW – ground_crew.h264



Canny Edge – gorund_crew.h264

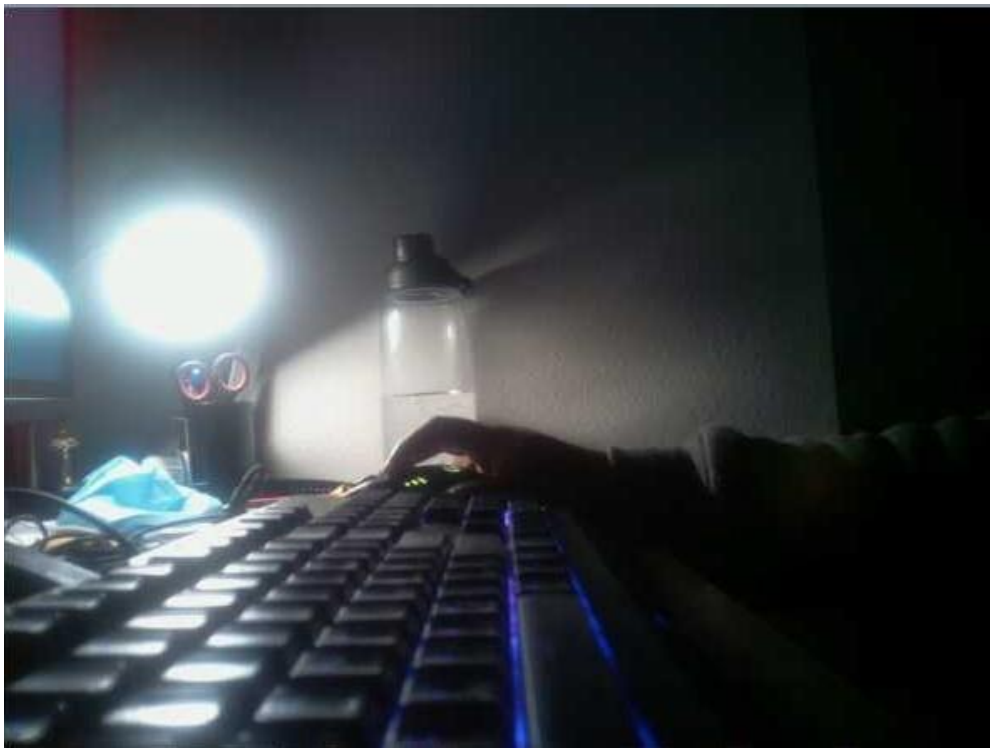


camera_canny_psnr.cpp execution for ground_crew.h264

Observations

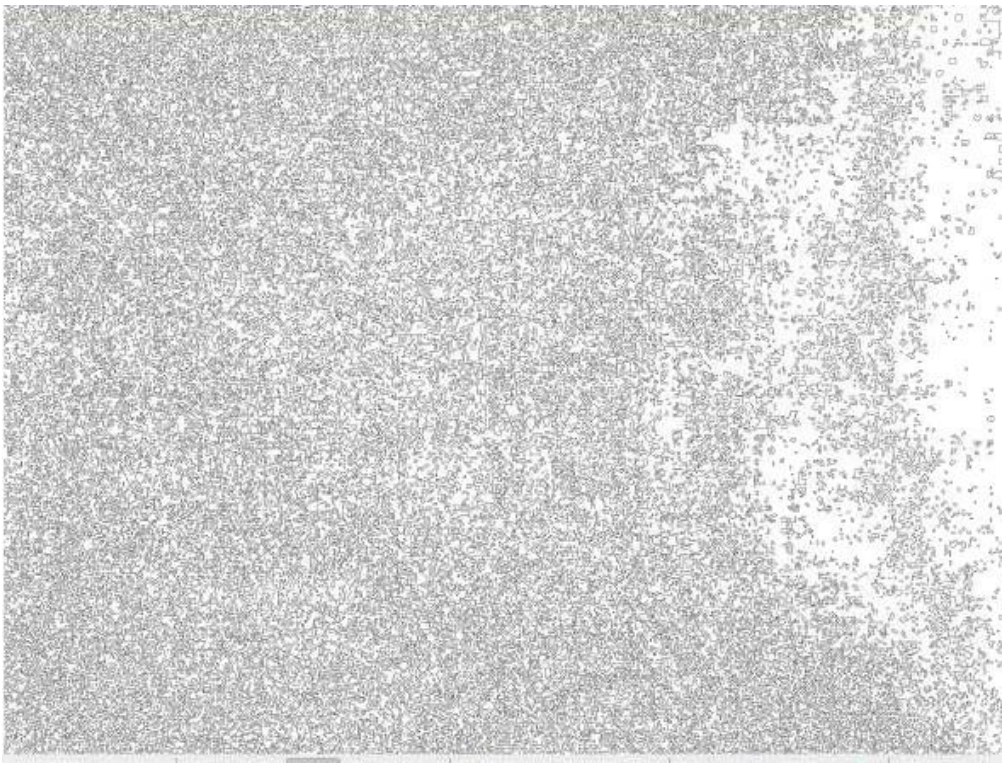
Trial	Sigma	Tlow	Thigh	FPS	Average PSNR
1	1.0	0.2	0.6	1.955208	7.97
2	0.5	0.5	0.5	2.032745	7.89
3	0.3	0.6	0.9	2.417985	7.95
4	0.1	0.1	0.1	2.303001	7.98
5	0.1	0.2	0.3	2.387405	7.97

- For RPi Camera (screen.jpg)



Webcam Image (RPi Camera)

RAW – Webcam Image



Canny Edge – Webcam Image

```
pi@raspberrypi:~ $ cd Desktop/204_hw4/camera
pi@raspberrypi:~/Desktop/204_hw4/camera $ g++ `pkg-config --cflags --libs opencv` -I. c
anny_util.c calcpsnr.c camera_canny_psnr_cam.cpp -o psnr
pi@raspberrypi:~/Desktop/204_hw4/camera $ ./psnr 1.0 0.2 0.6
Media Input: 2592, 1944
Sleep 3 seconds for camera stabilization...
=== Start Canny Edge Detection: 30 frames ===
=== Finish Canny Edge Detection ===
Total Elapsed Time : 100.530665 sec.
FPS: 0.298416.
Average PSNR value = 0.79
pi@raspberrypi:~/Desktop/204_hw4/camera $ ./psnr 0.5 0.5 0.5
Media Input: 2592, 1944
Sleep 3 seconds for camera stabilization...
=== Start Canny Edge Detection: 30 frames ===
=== Finish Canny Edge Detection ===
Total Elapsed Time : 92.602860 sec.
FPS: 0.323964.
Average PSNR value = 0.93
pi@raspberrypi:~/Desktop/204_hw4/camera $ ./psnr 0.3 0.6 0.9
Media Input: 2592, 1944
Sleep 3 seconds for camera stabilization...
=== Start Canny Edge Detection: 30 frames ===
=== Finish Canny Edge Detection ===
Total Elapsed Time : 79.304759 sec.
FPS: 0.378288.
Average PSNR value = 0.26
```

camera_canny_psnr.cpp execution for screen.jpg

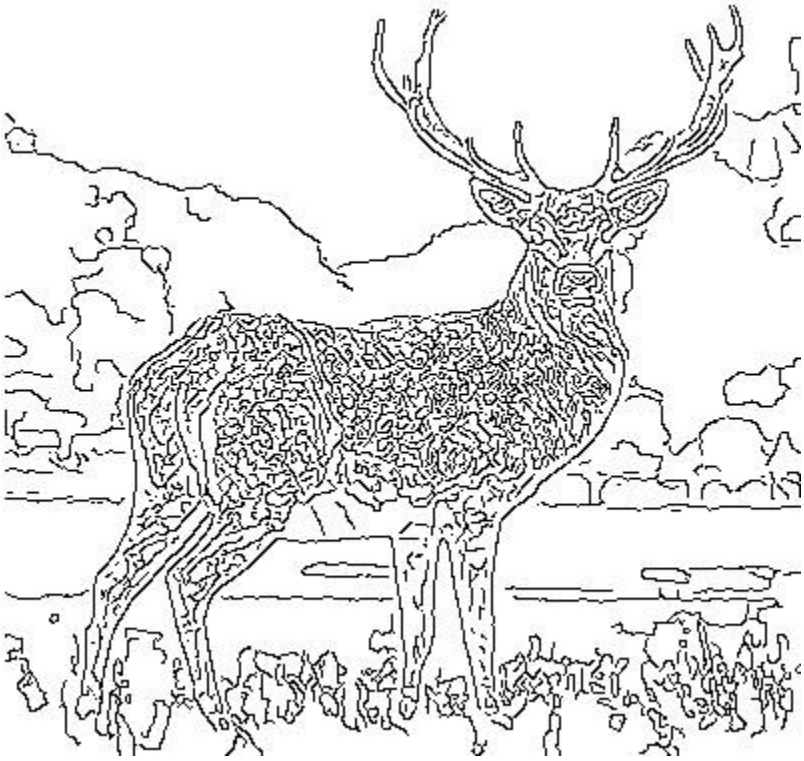
Observations

Trial	Sigma	Tlow	Thigh	FPS	Average PSNR
1	**1.0**	**0.2**	**0.6**	**0.2998416**	**0.79**
2	**0.5**	**0.5**	**0.5**	**0.323964**	**0.93**
3	**0.3**	**0.6**	**0.9**	**0.378288**	**0.26**
4	**0.2**	**0.4**	**0.1**	**0.380774**	**0.70**
5	**0.8**	**0.6**	**0.3**	**0.295851**	**0.94**

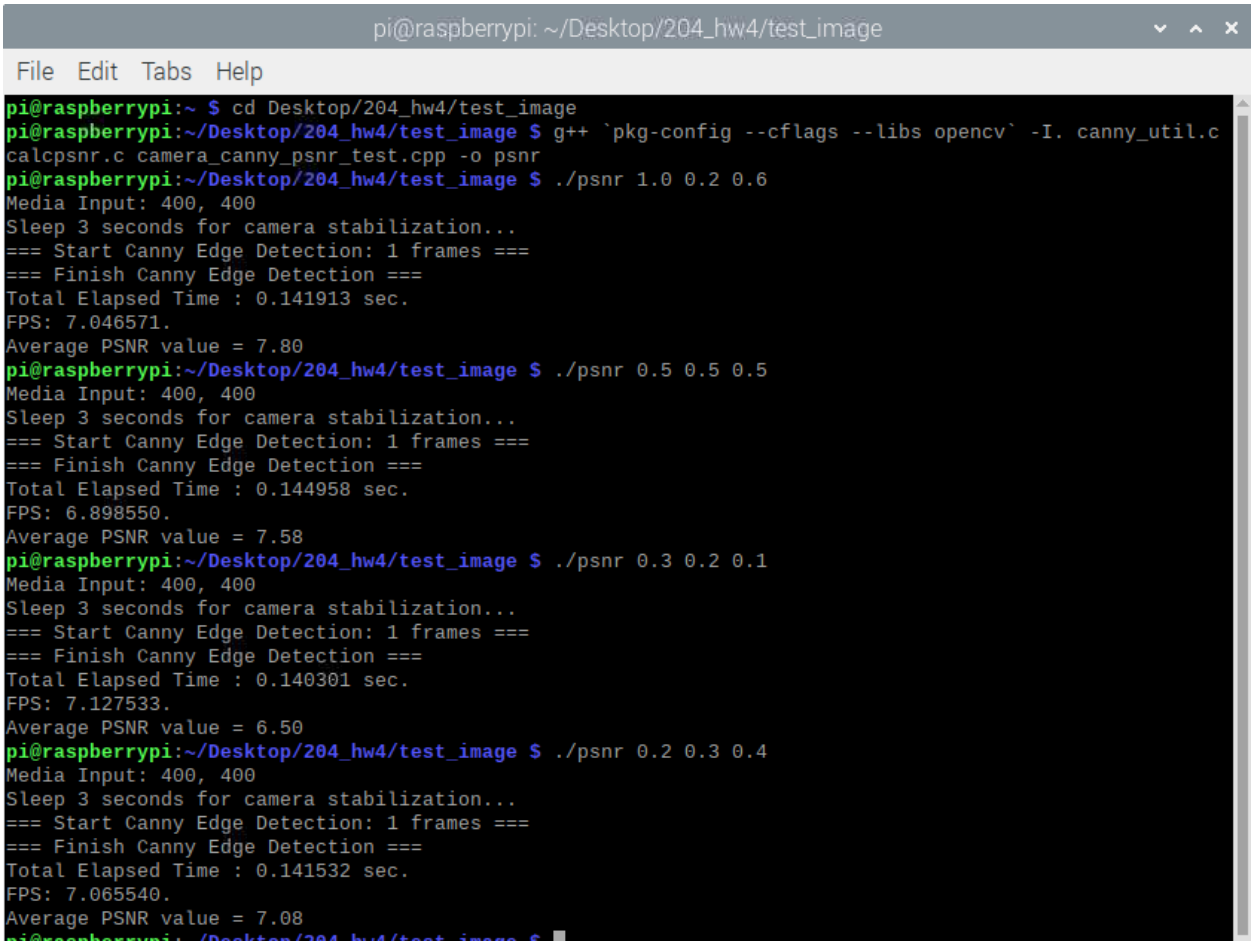
- For test image – deer.jpeg (downloaded from Internet) – 400x400

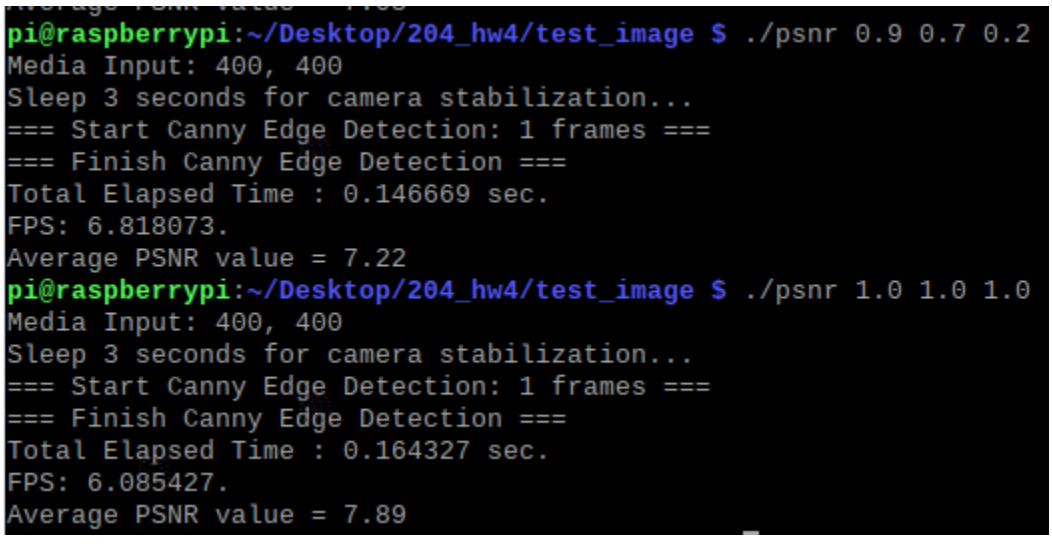
deer.jpeg (400 x 400)

RAW – deer.jpeg



Edge – deer.jpeg



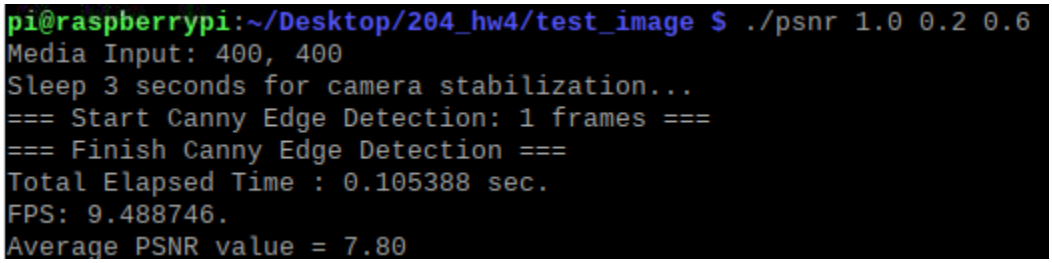


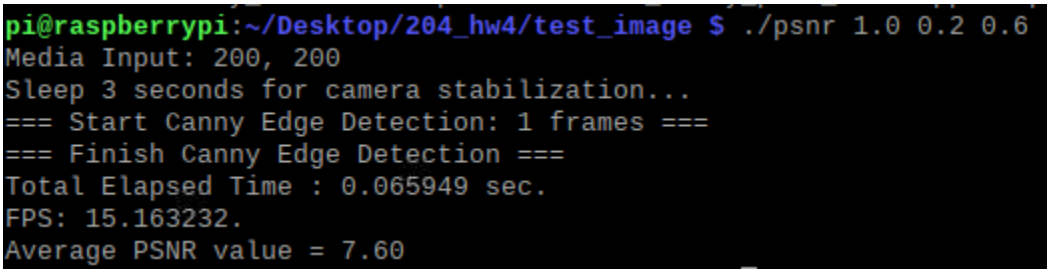
camera_canny_psnr.cpp execution for screen.jpg

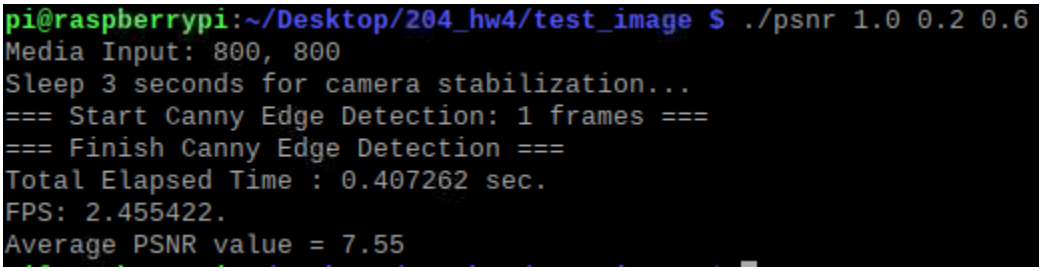
Observations

Trial	Sigma	Tlow	Thigh	FPS	Average PSNR
1	1.0	0.2	0.6	7.046571	7.80
2	0.5	0.5	0.5	6.898550	7.58
3	0.3	0.2	0.1	7.127533	6.50
4	0.2	0.3	0.4	7.065540	7.08
5	0.9	0.7	0.2	6.818073	7.22
6	1.0	1.0	1.0	6.085427	7.89

- Change in resolution for a fixed set of parameters – **1.0 0.2 0.6**







Observations

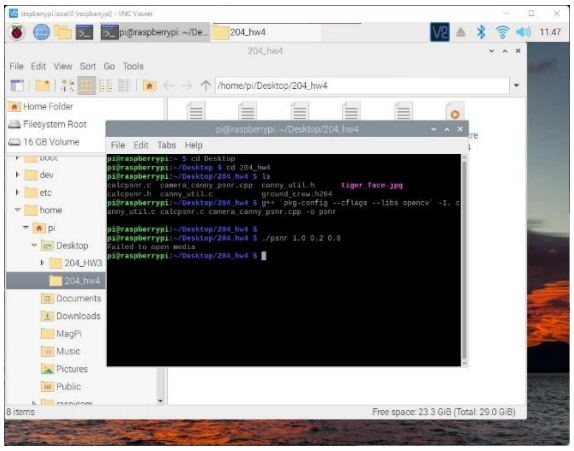
Resolution	FPS	Average PSNR
200 x 200	15.163232	7.60
400 x 400	9.488746	7.80
800 x 800	2.455422	7.55

- The Average PSNR doesn't offer much deviation for tiger_face.jpg and ground_crew.h264
- The Average PSNR greatly varies for the RPi camera observations – high variance.
- The elapsed time for ground_crew.h264 is vastly major as compared to tiger_face.jpg. Over 100 seconds as compared to the nominal 13-14 seconds. This may be due to the psnr executable processing a video file and the number of NFRAMES being more.
- The elapsed time for RPi Camera also has a high variance among execution trials ranging from 78-101 seconds.
- Edge quality degrades with extreme values for sigma, thigh and tlow since image appears over processed and overly bright/dark.
- The downloaded deer image was of type 'jpeg' which seems to work with PSNR.
- Upon processing the jpeg image, the PSNR executable also creates a jpg version titled 'jpg'
- The optimal parameters for tiger_face.jpg are **0.8 0.4 0.2**; **0.1 0.2 0.6** & **0.3 0.6 0.9**
- The optimal parameters for ground_crew.h264 are **0.1 0.1 0.1; 0.1 0.2 0.3** & **0.3 0.6 0.9**
- The optimal parameters for screen.jpg are **0.2 0.4 0.1**; **0.3 0.6 0.9** & **0.5 0.5 0.5**
- The optimal parameters for deer.jpg are **0.3 0.2 0.1**; **0.2 0.3 0.4** & **1.0 0.2 0.6**
- The worst parameter for tiger_face.jpg is **0.6 0.8 0.3**
- The worst parameter for ground_crew.h264 is **0.1 0.2 0.6**
- The worst parameter for screen.jpg is **0.8 0.6 0.3**
- The worst parameter for deer.jpg is **0.1 0.1 0.1**
- All parameters determined are from observed values **ONLY**
- For the RPi Camera output – screen.jpg, the optimal parameters are difficult to determine since either too high or too low values of sigma, tlow and thigh affect the edge detection and the final image is too hard to determine from the original input.

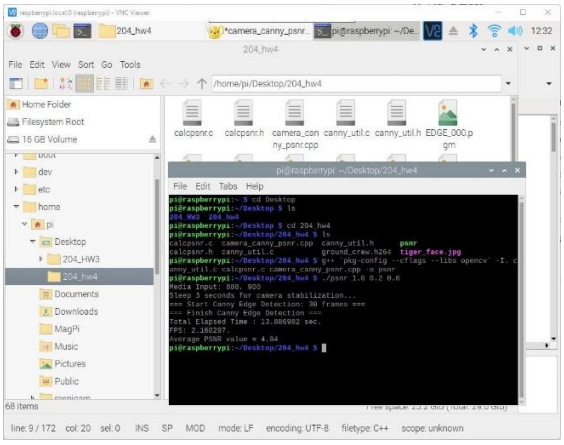
- Bad lighting greatly affects the output of FPS and PSNR generation since darker images on the Pi Camera resulted in worse outputs as compared to a brighter image.
- Bright lighting affects the outputs of FPS and PSNR generation as well since brighter images on the Pi Camera resulted in bad outputs as compared to a normally bright image.
- When resolution is halved, there's a large boost in FPS but slight drop in average PSNR.
- When the resolution is doubled, there's a large drop in FPS and a slightly larger drop in average PSNR.
- At times, the FPS cannot be improved further past a threshold since it would come at the cost of degrading image quality.

4. Problems and Discussion (6 pts)

- The PSNR executable constantly gave errors to input media since the camera_canny_psnr.cpp file wasn't edited yet.



Solution was to add an imread() function to camera_canny_psnr.cpp to generate proper psnr executable



- Finding util.c/h files was an issue since the Canvas page mentioned it being in Project#2. However, it was located on Project#3
- The RAW image output for ground_crew.h264 and screen.jpg consisted of black screens in its output and the edge outputs were as well.
- The results agree with my expectations since the Average PSNR generation took longer for ground_crew.h264 and tiger_face.jpg looks optimal without being overly saturated in the edge output while using minimal parameters.
- Holding the Pi Camera constant was an issue since screen.jpg always ended up being blurry and was difficult to process without getting a proper output.
- Solution was to keep it steady over a flat surface.
- Bad lighting while using the Pi camera added to the worse output apart from it being blurry.
- Solution was to take an image in a brighter environment.
- Facing the RPi Camera towards the light for a brighter image isn't optimal since it results to just a white spot in edge detection and also distorts object edges around it.
- Solution is to not face the light but on an object under it.
- There was an OpenCV Exclusion error on the first time while processing the deer.jpeg image downloaded from the Internet. It didn't show up the next time.
- Slight variations occur in FPS and average PSNR upon doubling or halving the resolution of the image.
- This project was extensively long which resulted in degradation of values generated overtime due to heating of the RPi board.
- It could be improved by executing the project in min-parts rather than a whole.
- Getting results could be efficiently done by just changing the sigma, tlow, thigh values for a given input.
- The canny_util.c/h files could've been included within Canvas for easy access.

5. Conclusion (2 pts)

- From this project, one can conclude that the PSNR is a form of error calculation that utilizes the mean squared error of pixel values by comparing two images. The higher the PSNR, the better degraded image ahs been reconstructed to match the original image. This is because we wish to minimize the MSE between images with respect to the maximum signal value of the image.

- The best parameters aren’t always constant and depend on the lighting and saturation of the image being processed and its resolution along with other underlying parameters.

6. Sources

- https://www.howtogeek.com/109369/how-to-quickly-resize-convert-modify-images-from-the-linux-terminal/
- https://www.ni.com/en-us/innovations/white-papers/11/peak-signal-to-noise-ratio-as-an-image-quality-metric.html
- https://meridianthemes.net/jpg-vs-jpeg-file-formats/