ECE 437

Spring 2020

Final Project

Realtime Video and Temperature Acquisition

Haofan Lu
Yiqing Xie
Section: AB1/ Tuesday 9:00 am-11:50am

- Introduction

In this final project, we designed and implemented a real time video and temperature acquisition system which is capable of acquiring image data from CVM300 image sensor with frame rate 20fps and acquiring temperature data from ADT7420 temperature sensor with at least 100 readings per second. We display the result as a real time video with temperature reading on it.

- Description of algorithm

The implementation of this project includes both software and hardware. For software, we use Python and its libraries to take care of the configuration and display logic. In particular, we used the Opal Kelly library to communicate with the FPGA. We used opencv library to display the acquired data as video.

For the configuration logic, we listed the register addresses and values as a dictionary and wrote write and read functions to send the values of each register to the FPGA through wires in OK library. To ensure the imager is programmed correctly, we double checked the values of registers by reading back the values and comparing with the setting value. If the imager is not validly programmed, we stop the procedure.

The python code also takes care of the data processing and display. The temperature data we acquired from the temperature sensor is of 16-bit resolution, we wrote a function to convert it to the decimal value in Celsius degree and display it on the video through opencv putText. The frame rate and temperature reading rate are both calculated and displayed on the video.

On the hardware side, we used I2C protocol to interface with the temperature sensor, we wrote a FSM to take care of the signal sequence. The temperature sensor returns data with. We sent this data to python via OK wireout module. For image data, we used a 32-bit in/32-bit out FIFO module to deal with the I/O with different clock. We choose to deal with the byte inversion algorithm in hardware, because it is faster than coping with it in python.

- Results and discussion

We collected data of pixel [50,50] with different exposure time and compute its temporal noise, signal to noise ratio and conversion gain

1. Temporal noise of the image sensor

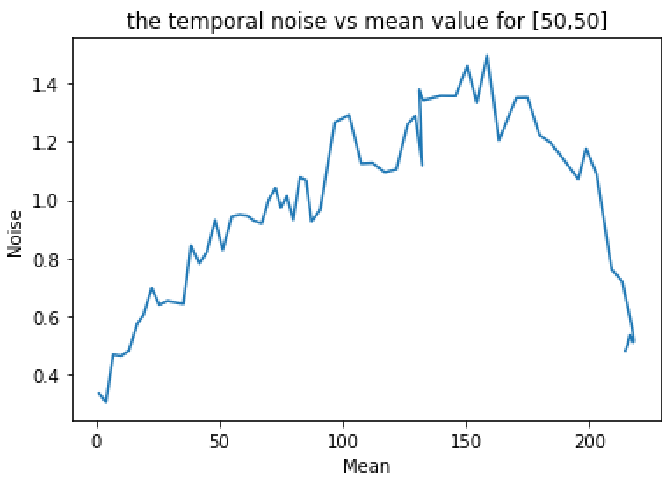

Figure 1 temporal noise vs mean value

Figure 1 shows the relation between signal intensity and temporal noise. As exposure time increases, the signal intensity increases, and the temporal noise increases. But when the imager reaches saturation, the temporal noise decreases.

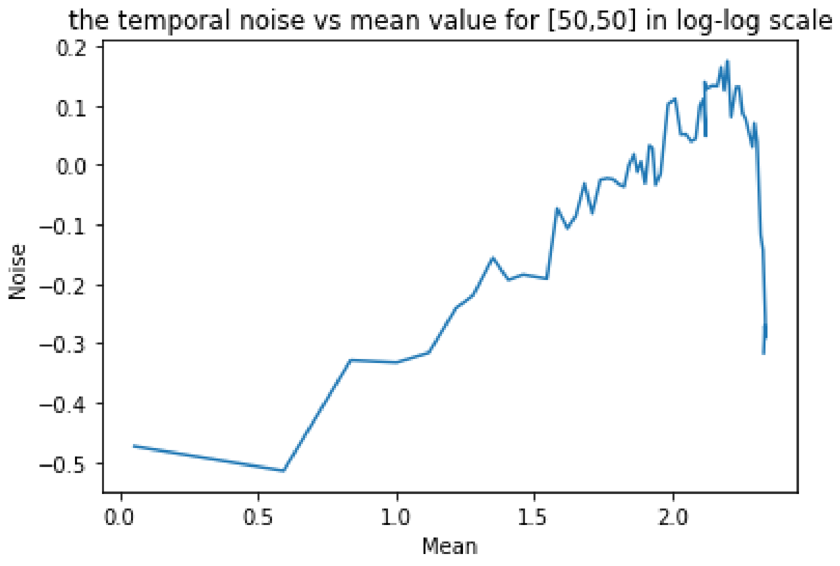

Figure 2 temporal noise vs mean value (log-log)

Figure 2 is the plot of temporal noise vs. mean value in log-log scale.

2. Signal to noise ratio of the sensor

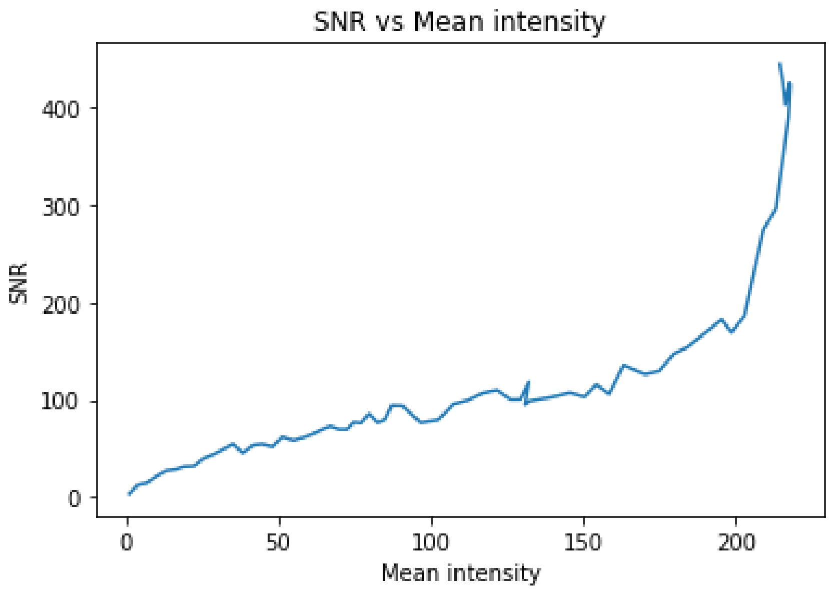

Figure 3 signal to noise ratio

The SNR behaves as a polynomial with index lower than 1 in range of 0 to 200. Once it reaches 200, it saturates and the SNR increases quickly.

3. conversion gain of the sensor

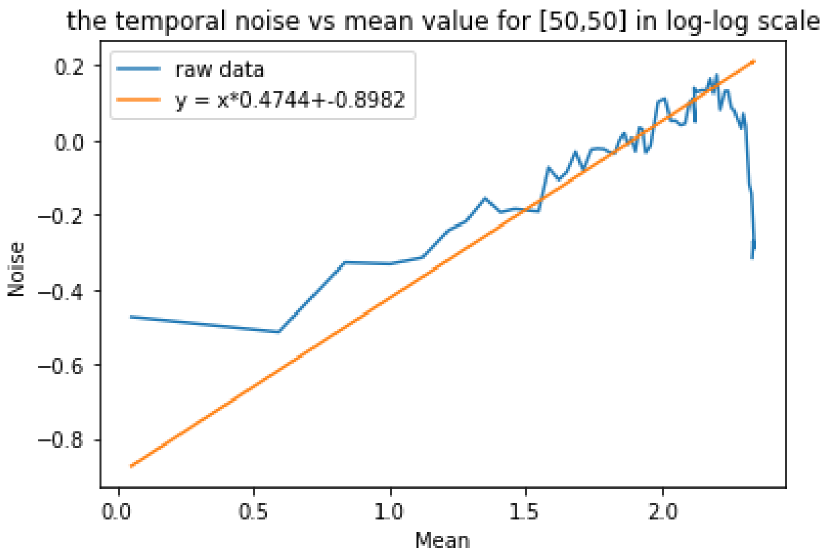

Figure 4 temporal noise vs mean plot in log-log scale with regression line.

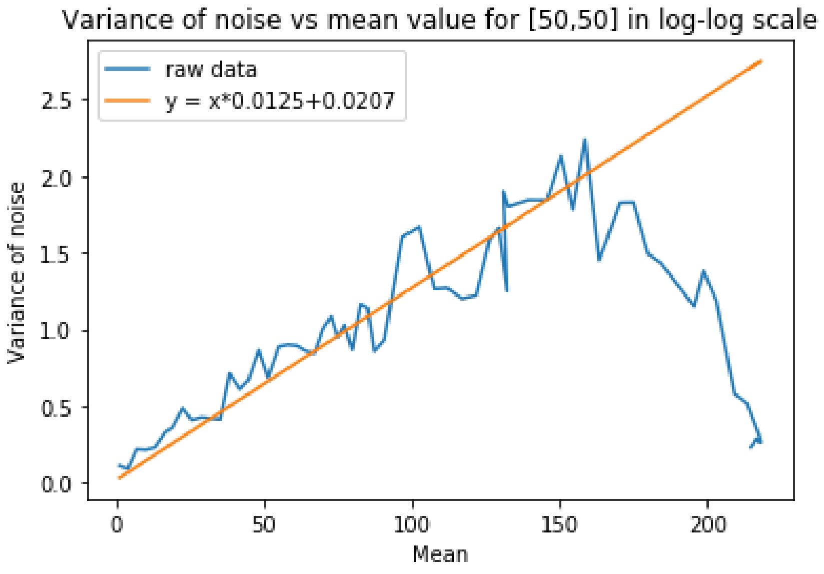

Figure 5. find conversion gain by variance method

The slope of the regression line is 0.4744 in log-log plot. The conversion gain for the pixel [50,50] is 0.0128 DN/e- (78.125 e-/DN) by the mean-standard deviation method. And 0.0125 DN/e- (80 e-/DN) by the mean-variance method.

4. noise in the temperature sensor

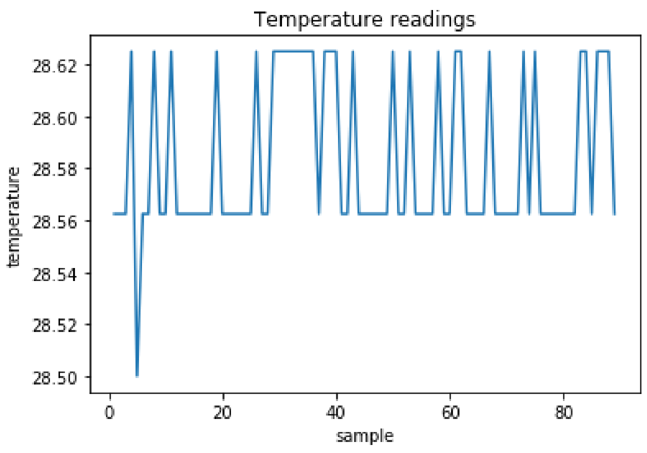

Figure 6. temperature readings

The noise (standard deviation) of the temperature sensor readings is 0.03075

5. limitations

The main issue that limits our data acquisition system is the block size and the software speed. We used to try to invert the byte sequence in python, but it is too slow in python, we finally chose to implement the inversion process in Verilog. To achieve higher frame rate, we chose to use the multiframe mode of the imager. In this way we can use greater block size to achieve higher data transfer rate. However, it is trade off, because larger block takes longer time to finish transferring, which can make the video glitch. To make the video more natural, we finally chose the frame number to be 4 frames per block. In this way, we can use the block size of 256 bytes.

- Conclusion

The final project integrates and further develops everything we have learned through this semester. We use okWire to transmit commands from software to hardware, and temperature data from hardware to software. We also use Block-Throttled Pipe to deal with image data. There major progress in this project is that we successfully read image continuously from the camera and display images using opencv. The final video window shows the camera images with texts indicating frame rate and temperature data. I believe this is a great work which meets all requirements.