IRL #2: Progress Review

Man-Ning Chen (Mandy)

Team G: EXCALIBR

Teammates: Yiqing Cai, Huan-Yang Chang,

Siddharth Raina, Sambuddha Sarkar

Individual Progress

Overview

We are now using Aerotech robotic arm and Emergent Vision Technologies cameras to build up a prototype of our future calibration system. I am responsible for camera setups including EVT API implementation and CMOS sensor calibration.

CMOS calibration

Before geometric, photometric and light-field calibrations, CMOS calibration is required. The sensor we are dealing with is fixed pattern noise. There are two kinds of fixed pattern noises : Dark Signal Non-Uniformity (gain FPN) and Pixel Response Non-Uniformity (offset FPN). The relation between pixel value and FPN can be written as follows,

$$y = ax + b$$

Measured intensity value — Gain FPN — True intensity value — Offset intensity value

Offset FPN is resulted from temperature and it exists even when there's no light source. Therefore it can be easily resolved by simply taking pictures when the camera is in dark. (when x=0, y = b).

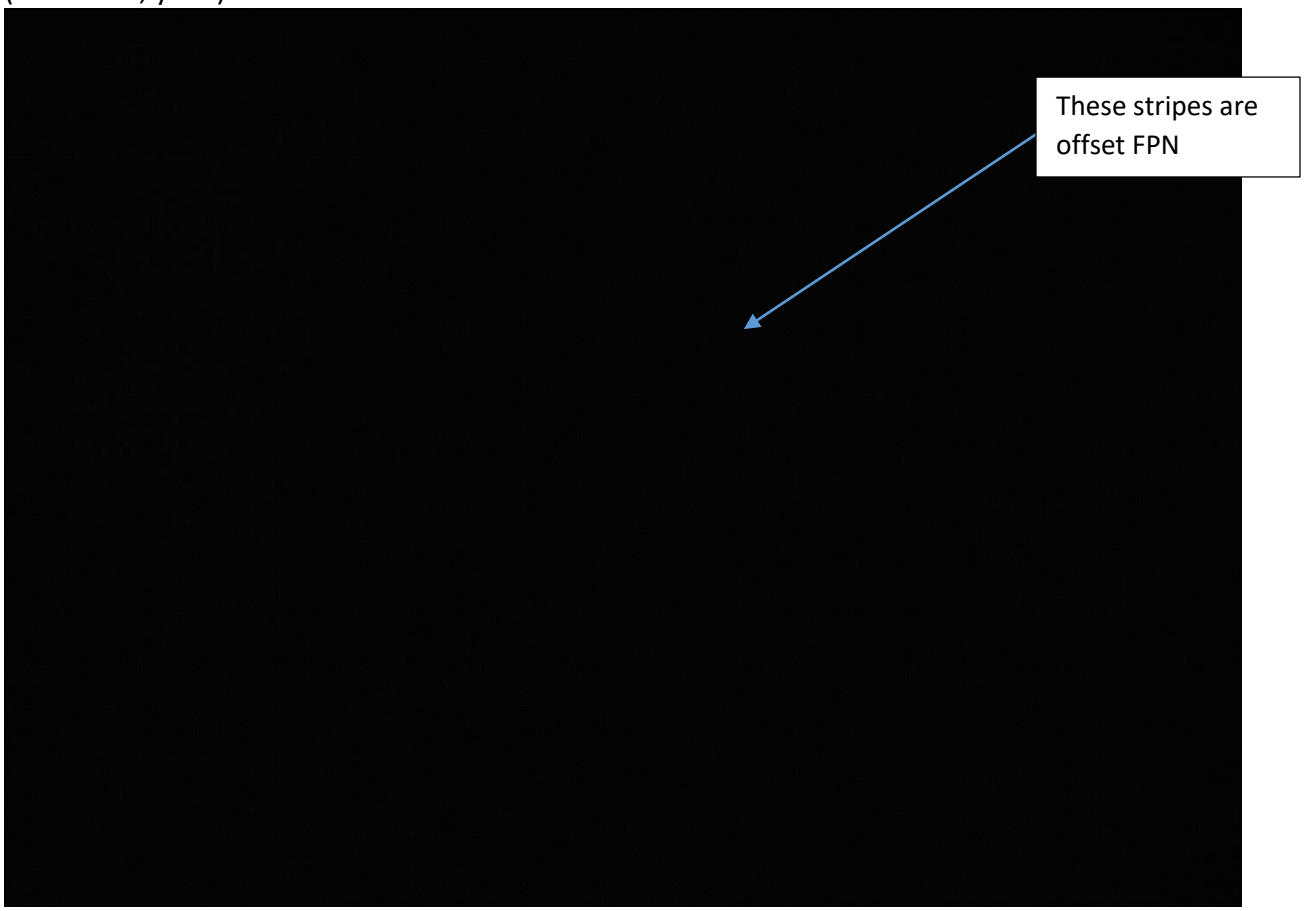
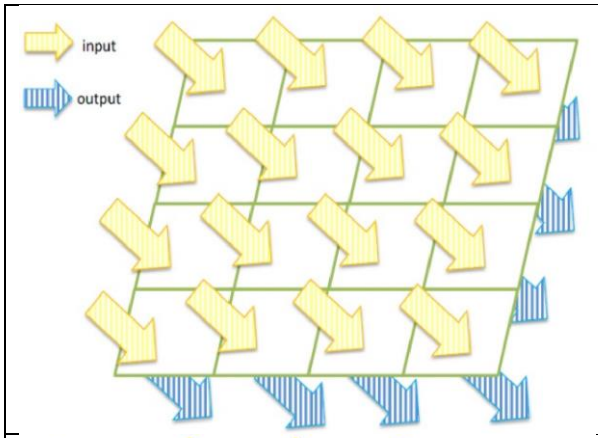
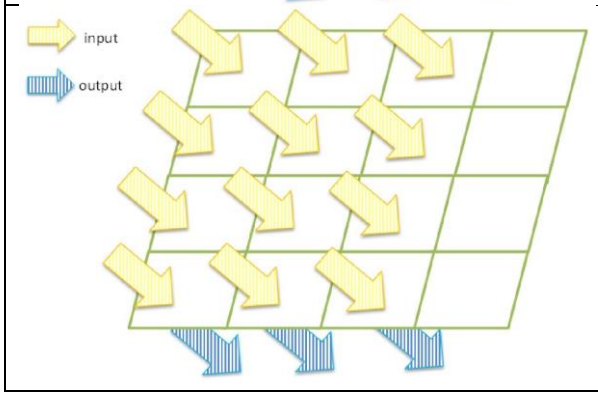


Figure 1. Photos taken in dark

On the other hand, gain FPN is more difficult to get because it is coupled with the true intensity value. I studied several papers [1] [3] Mohammadnejad, Shahram, Sobhan Roshani, and Saeed Roshani. "A Novel Fixed Pattern Noise Reduction Technique in Image Sensors for Satellite Applications." Electrical and Electronic Engineering 2.5 (2012): 271-276.
and decided to implement Mohammadnejad's method [3] Mohammadnejad, Shahram, Sobhan Roshani, and Saeed Roshani. "A Novel Fixed Pattern Noise Reduction Technique in Image Sensors for Satellite Applications." Electrical and Electronic Engineering 2.5 (2012): 271-276.
.



$$y_1(1,1) = a(1,1)x(1,1) + b(1,1)$$
$$y_1(1,2) = a(1,2)x(1,2) + b(1,2)$$
$$y_1(1,3) = a(1,3)x(1,3) + b(1,3)$$
$$y_1(1,4) = a(1,4)x(1,4) + b(1,4)$$



$$y_2(1,1) = a(1,1)x(1,2) + b(1,1)$$
$$y_2(1,2) = a(1,2)x(1,3) + b(1,2)$$
$$y_2(1,3) = a(1,3)x(1,4) + b(1,3)$$

Table 1 shows the relation between frames and pixel values
With three consecutive photos shifted by one pixel from one photo to its next photo, we get the relation of pixel values and FPN:

$$y(1,1) = a(1,1)x(1,1) + b(1,1)$$
$$y(1,2) \times f_1(1,2) = a(1,1)x(1,2) + b(1,2) \times f_1(1,2)$$
$$y(1,3) \times f_1(1,3) = a(1,1)x(1,3) + b(1,3) \times f_1(1,3)$$
$$y(1,4) \times f_1(1,4) = a(1,1)x(1,4) + b(1,4) \times f_1(1,4)$$

y, f, b are known while a and x are unknown.
It can be seen that if we set a(1,1)*x as the true intensity value, then the new true intensity value is just a scaled value of the original true value. Therefore, we can get the gain FPN relation between pixels and calibrate the CMOS.

However, in reality, we cannot move our camera by only one pixel. Hence, we put apriltag on a white paper and track the apriltag to learn how many pixels the camera move.

Figure 2 Apriltag

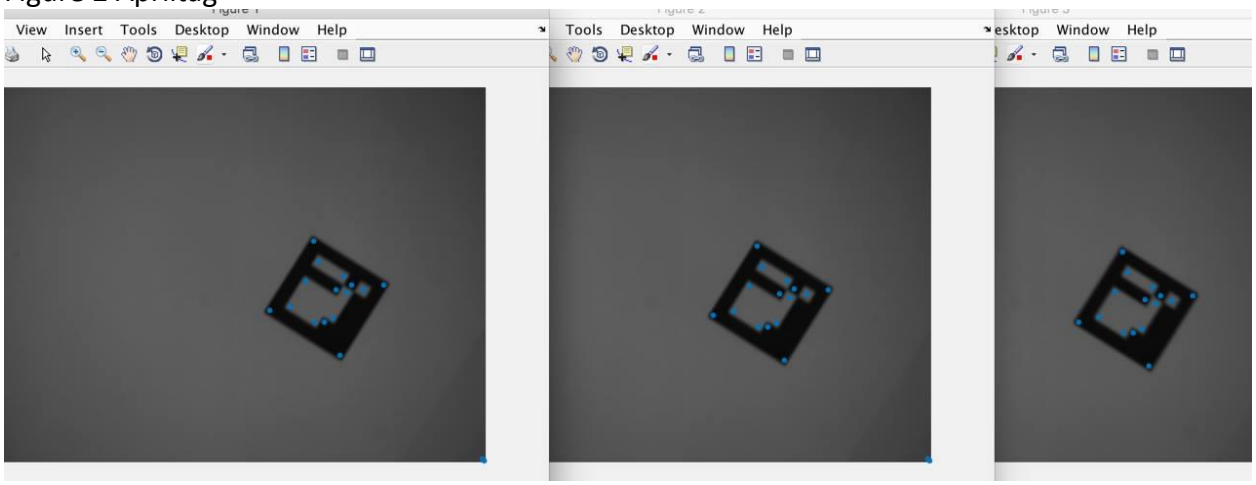


Figure 3 Detected corners of the apriltag

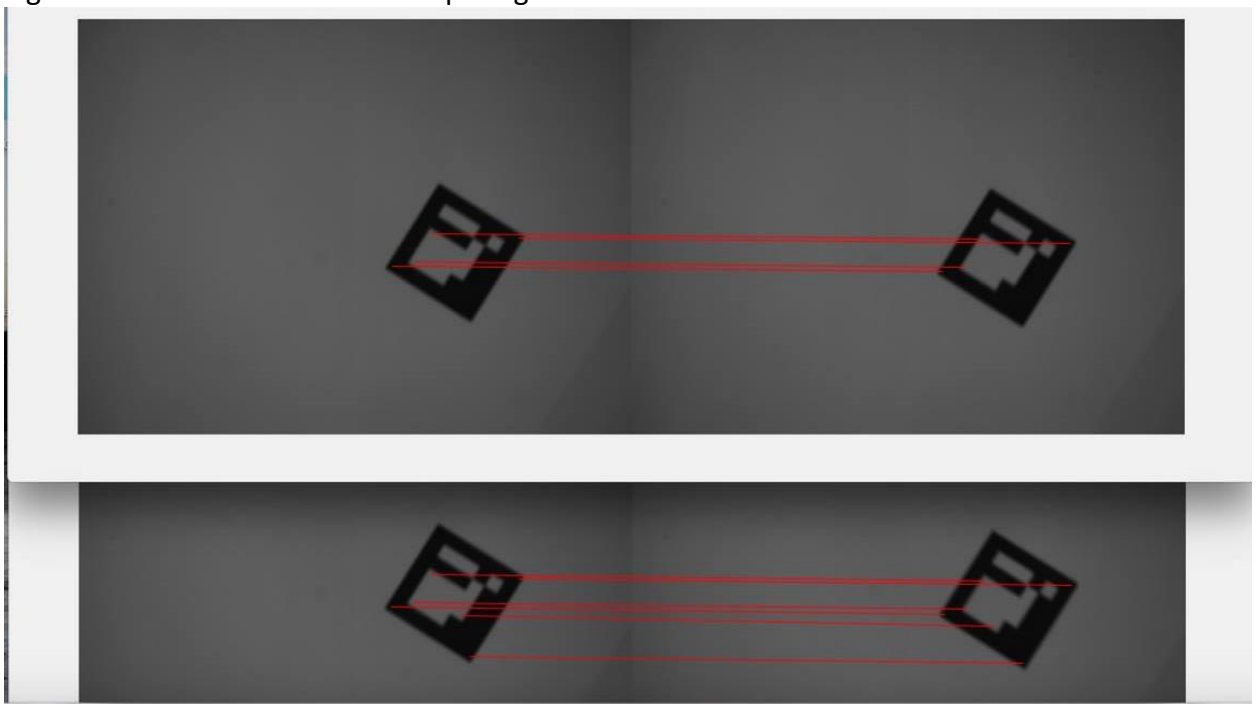


Figure 4 Tracking corners

Then, we can modify the algorithm provided by Mohammandnejad[3]. We will change it from one pixel movement to several pixels (the number derived from tracking corners) movement.

Figure 5 EVT camera

In addition to this algorithm, I also implemented EVT camera API, made it possible to trigger the camera, streamed images from camera to the computer and stored images in the computer.

Figure 6 Pictures taken by API

Challenges

We have encountered a problem that we could not trigger multi-cameras at the same time. We implement EVT API in our code (on visual studio platform) and change it to make multiple camera open, stream and record. However, we found that every time when we

stop our program while it was running or it stop itself because of bugs in the program, we would face failure reopening the cameras next time. At first we thought we might be using the API codes wrong. After several tests, we found out that cameras needed to be closed by explicit API commands and since our commands for closing camera is put at the last part of our code, the program wouldn't execute the command if it stop before the command. Our resolution is to make sure the camera is closed before we run the program. There are two ways to achieve this:

1. Run the close command.

2. Restart the cameras.

Teamwork

Yiqing Cai	Algorithm development, camera setup
Huan-Yang Chang	Robot arm control & calibration target fabrication
Siddharth Raina	Camera setup
Sambuddha Sarkar	Robot arm setup

Future Plan

Currently, Harris corner detector doesn't work on our input images. It is probably because the resolution is too high and the image is too big. For a very big image, we should use a larger kernel when implementing Harris corner detector. We will modify our detector kernel in the future.

Furthermore, the algorithm for shifting frame by multiple pixels is still in progress. We'll keep developing it and then implement it on the images.

We will try enlarge kernel size of the detector or other detection method.References

[1] Toczek, Tomasz, et al. "Scene-based non-uniformity correction: from algorithm to implementation on a smart camera." *Journal of Systems Architecture* 59.10 (2013): 833-846.
[2] Harris, John G., and Yu-Ming Chiang. "Nonuniformity correction using the constant-statistics constraint: analog and digital implementations."*AeroSense'97*. International Society for Optics and Photonics, 1997.
[3] Mohammadnejad, Shahram, Sobhan Roshani, and Saeed Roshani. "A Novel Fixed Pattern Noise Reduction Technique in Image Sensors for Satellite Applications." Electrical and Electronic Engineering 2.5 (2012): 271-276.