浙江大學

Lab 1 Report

Group 3

Group Members and IDs:

Pan Yue（3220100912）

Wang Haoyu（3220105740）

Hu Jiyuan（3220104116）

Date: September 14_{th}, 2024

Teacher: Wu Hongzhi

TA : Fei Fan

Table of Contents

1 Introduction

Study the relationship between camera parameters and captured images(pixels) via experiments:

1. Exposure
2. Gain

2 Experiment Setup

2.1 Camera Specifications

Here is some basic informations about the camera used in lab.

- Dual Basler Dart Machine Vision USB3 Color Cameras + Lens
- Resolution: 2592x1944
- 14fps
- Aperture: F1.6
- Focal Length: 8.0mm
- Adjustable Baseline

2.2 Turning Off Automatic Features

Before we start, we need to turn off all automatic features of the camera. Three features are involved:

- White balance
- Exposure
- Gain

Turn off these features by editing the `SetCamera` function:

```
def SetCamera(camera,cameraSettings):
    #set whitebalance
    camera.PixelFormat.SetValue(cameraSettings['PixelFormat'])
    camera.UserSetSelector = "Default"
    camera.UserSetLoad.Execute()
    camera.BalanceWhiteAuto.SetValue("Off")
    camera.BalanceRatioSelector.SetValue = "Red"
    camera.BalanceRatio.SetValue(cameraSettings['r_balance'])
    camera.BalanceRatioSelector.SetValue = "Green"
```

```
        camera.BalanceRatio.SetValue(cameraSettings['g_balance'])
        camera.BalanceRatioSelector.SetValue = "Blue"
        camera.BalanceRatio.SetValue(cameraSettings['b_balance'])
        #exposure time us
        camera.ExposureTime.SetValue(cameraSettings['exposure_time'])
        camera.Gain.Value=cameraSettings['gain_db']
        camera.GainAuto.SetValue("Off")
        camera.ExposureAuto.SetValue("Off")
```

2.3 Obtaining Images

We obtained pictures of different exposure and gain by taking two `for` loops:

```
while camera.IsGrabbing():
    for exposure_time in range(200, 10000, 200):
        for gain in range(0, 1001, 10):
            # Upadate exposure and Gain
            print(f"gain: {gain}")
            camera.Gain.Value = 1
            camera.ExposureTime.SetValue(exposure_time)
            # Wait for an image and then retrieve it. A timeout of
               5000 ms is used.
            grabResult = camera.RetrieveResult(5000, pylon.
               TimeoutHandling_ThrowException)
            k = cv2.waitKey(1000)
            if grabResult.GrabSucceeded():
                # Access the image data.
                Converter = pylon.ImageFormatConverter()
                Converter.OutputPixelFormat = pylon.
                   PixelType_BGR8packed
                Converter.OutputBitAlignment = pylon.
                   OutputBitAlignment_MsbAligned
                img = Converter.Convert(grabResult)
                img = img.GetArray()
                img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  # 转换颜色
                   空间从BGR到
                cv2.imshow('Grabbed Image', cv2.resize(img, None, None
                   , fx=0.2, fy=0.2))

                # if(k==ord('s') or k==ord('S')):
                os.makedirs(save_root, exist_ok=True)
                # filename = f"{save_root}/exposure_{exposure_time}_us
```

```
                .png"
            filename = f"{save_root}/noise_{image_count}.png"
            Img.AttachGrabResultBuffer(grabResult)
            Img.Save(pylon.ImageFileFormat_Png, filename)
            print(f"save to {filename}")
            image_count += 1
            print(f"{image_count}")
            grabResult.Release()
            # if(k==ord('q') or k==ord('Q')):
            #     print(f"save to {save_root}")
            #     break
        else:
            print("Error: ", grabResult.ErrorCode, grabResult.
                ErrorDescription)
            grabResult.Release()
```

The second and the third line of code determines the different values of gain and exposure we are taken. during each loop, we save an PNG image using the `pylon.ImageFileFormat_Png` format. Each image's name is generated based on the current image count.

We may comment out one `for` loop (either the exposure or gain) to get data, since we only study one variable at one time.

3 Result and Data Processing

We process the images using the following two pieces of code to analyze the impact of exposure and gain on the image.

3.1 Exposure

```
from PIL import Image
import pandas as pd
import matplotlib.pyplot as plt

start = 2000
end = 98000

def exposure(pixel):
        red, green, blue = pixel

        red = int(red)
        green = int(green)
        blue = int(blue)
```

```
            result = pow(red/255, 2.2) + pow(green/255, 2.2) + pow(
                blue/255, 2.2)

            light = 0.547373 * pow(result, 1/2.2)

            return light

    exposures = []
    lights = []

    for i in range(start, end, 2000):
        file_name = f'exposure/exposure_2000_98000/exposure_{i}_us.png
            '
        image = Image.open(file_name)

        image = image.convert('RGB')

        width, height = image.size

        pixel = image.getpixel((1296, 972))
        light = exposure(pixel)

        exposures.append(i)
        lights.append(light)

        print(f"exposure time = {i} 的值: {light}")


    plt.scatter(exposures, lights)

    plt.title('Sample Line Plot')
    plt.xlabel('X Axis')
    plt.ylabel('Y Axis')
    plt.show()
```

This code generates a scatter plot that shows the relationship between different exposure times and the corresponding light values at a specific pixel location (1296, 972) in a series of images.

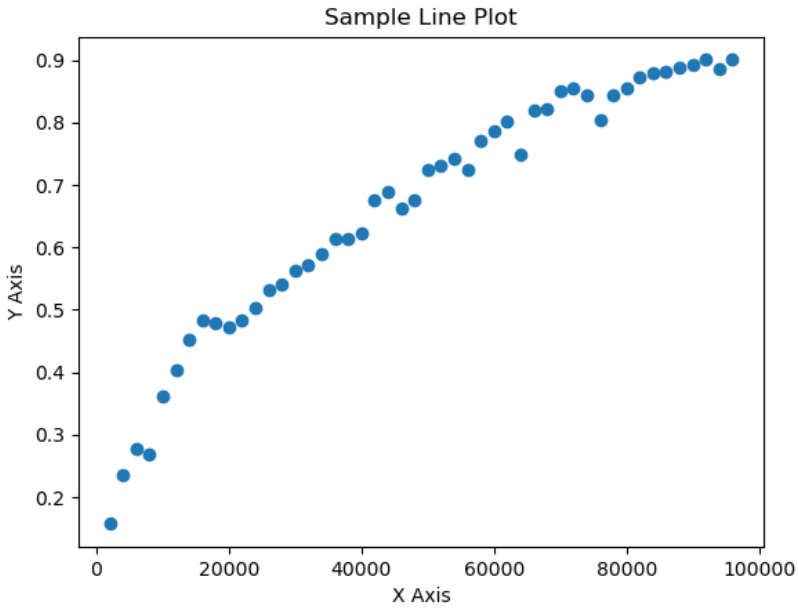

图 1: Exposure vs. Light Value

The resulting scatter plot visualizes how the light value at the specified pixel changes with different exposure times.

3.2 Gain

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# 读取图像
gain_images = []
gains = []
for i in range(0, 98):
    i = i/10
    image = cv2.imread(f'gain/gain_{i}.png', cv2.IMREAD_GRAYSCALE)
        # 加载为灰度图
    image = image.astype(np.float32)  # 转换为浮点型以便后续处理
    gain_images.append(image)
    gains.append(i)

# 计算每个增益下的图像像素均值
mean_pixel_values = [np.mean(gain_image) for gain_image in
   gain_images]
```

```
# 绘制增益与均值像素值的关系
plt.figure(figsize=(6, 4))
plt.plot(gains, mean_pixel_values, marker='o')
plt.title('Gain vs. Mean Pixel Value')
plt.xlabel('Gain')
plt.ylabel('Mean Pixel Value')
plt.grid(True)
plt.show()
```

This code generates a line plot that shows the relationship between different gain values and the corresponding mean pixel values of images.

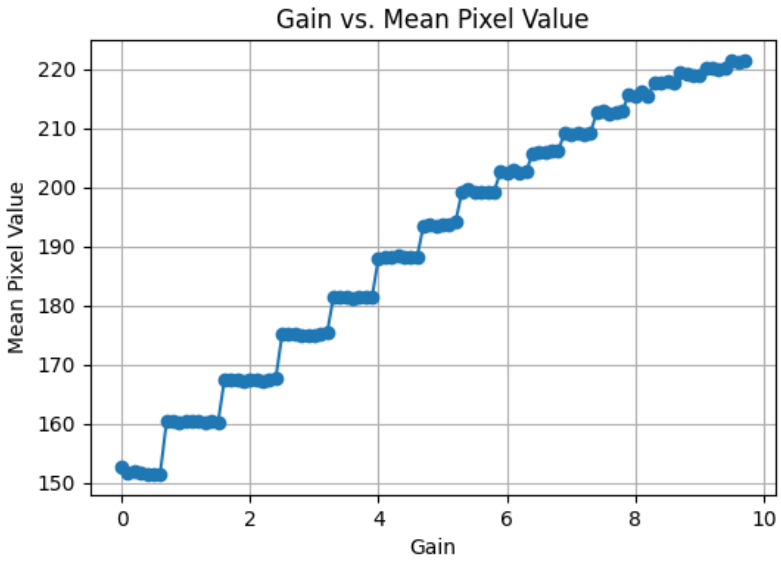

图 2: Gain vs. Mean Pixel Value

The resulting plot visualizes how the mean pixel value of the images changes with different gain settings.

4 Analysis and Discussion

Analyze with chart, formula and so on.

图 3: Images taken in different exposures (Increase from left to right)

1. **Exposure vs. Light Value**:

- The scatter plot of exposure times versus light values shows how the light intensity at a specific pixel changes with varying exposure times. As the exposure time increases, the light value generally increases, indicating that longer exposure times result in brighter images. Also, the higher the exposure, the slower the light value increases.

2. **Gain vs. Mean Pixel Value**:

- The line plot of gain values versus mean pixel values illustrates how the overall brightness of the image changes with different gain settings. As the gain increases, the mean pixel value also increases, indicating that higher gain settings amplify the image signal, resulting in brighter images. It is worth noting that the increase of the mean pixel value exhibit a staircase-like growth.

  This phenomenon may be due to the internal implementation of the gain settings. Gain typically increases brightness by amplifying the image signal, but this amplification is not linear and is instead segmented. Here are some possible reasons:

  - **Gain Stepping:** The camera's gain settings might be segmented rather than continuous. Each time the gain value reaches a certain threshold, the gain suddenly increases by a larger step and then remains stable within that step. This causes the mean pixel value of the image to exhibit a staircase-like growth.
  - **Gain Algorithm:** The camera's internal gain algorithm might handle different gain value ranges differently. For example, in the low gain range, the gain might be linear, while in the high gain range, the gain might be exponential or segmented linear.
  - **Hardware Limitation:** The camera hardware might limit the adjustment range and step size of the gain, causing the gain value to be adjusted in larger steps within certain ranges.

5 Conclusion

In conclusion, the experiments conducted in this lab have provided valuable insights into the relationship between camera parameters and the captured images. By varying the exposure time and gain settings, we were able to observe how these parameters affect the light values and pixel intensities in the images.

The results showed that longer exposure times result in brighter images with higher light values, while higher gain settings amplify the image signal, leading to increased pixel intensities. The increase of the mean pixel value exhibit a staircase-like growth. These observations provide a better understanding of how camera parameters influence the quality and brightness of the captured images.

Bonus Part

Introduction

In photography and image processing, a noise model is used to represent the random variations or disturbances that occur when capturing an image, affecting its quality. These noise models simulate or describe the kinds of imperfections introduced by various factors during the process of shooting a picture, such as the camera sensor, lighting conditions, and environmental factors.

The following part will introduce how we found out which noise model our camera system belongs to through experiments.

Experiment steps

1. While keeping the camera parameters and position the same, we took 50 photos of the same scene.

```
for img in image_list:
    accumulated_image += img.astype(np.float32)
```

2. Once all images are summed, we divide the accumulated pixel values by N (the total number of images) to calculate the average image.The average image represents the ”true” scene without noise.

```
average_image = accumulated_image / N
```

3. Compute Residual Images (Difference from the Average).We compute the residuals for each image by subtracting the average image from each individual image. These residual images represent the noise or differences between the actual image and the noise-free average image.

```
residual_images = []
for img in image_list:
    residual_image = (img.astype(np.float32) - average_image)
    residual_images.append(residual_image)
```

4. Since we are interested in analyzing noise intensity, we convert each residual image to grayscale. This simplifies the analysis by reducing the three color channels (RGB) to a single intensity channel.

```
residual_grayscale_images = [cv2.cvtColor(residual_image.astype(np
   .float32), cv2.COLOR_BGR2GRAY)  for residual_image in
   residual_images]
```

5. We flatten all the grayscale residual images into 1D vectors and concatenate them into a single array, allowing us to analyse the distribution of residual pixel intensity.

```
residual_values = np.concatenate([residual_image.flatten() for
    residual_image in residual_grayscale_images])
```

Result Analysis

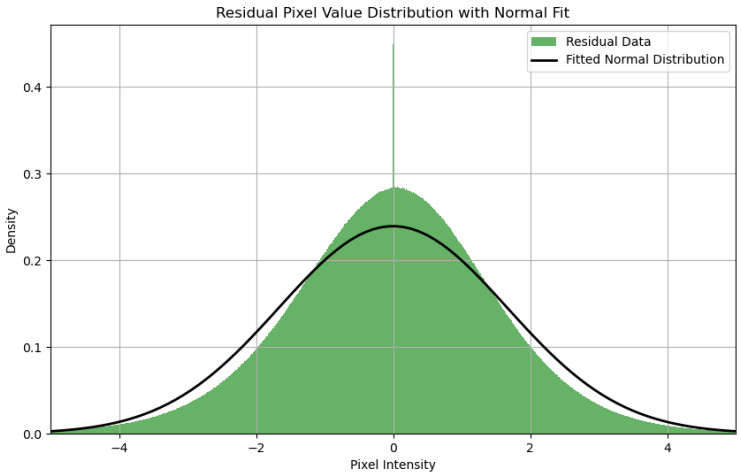

图 4: Gaussian noise

The distribution of Residual intensity roughly satisfies the normal distribution with a mean of 0, and the normal curve obtained by python fitting is also in good agreement with the actual data. It can be concluded that the noise model of our camera system is Gaussian noise.

- mean: -3.5373182072362397e-07
- standard deviation: 1.6672816276550293