Lab Report for

CE6023 – Computer Vision Systems

Student Name : **Dylan Rodrigues**

Student ID : **24121479**

Revision Timestamp: 17/11/2024 16:51:03

Contents

Introduction 2
Prepare and test the Slanted Edge Method..2
Experimental Setup 3
Data Collection for Variations of Camera Focus 4
Computer Vision & Measurements Procedures 6
Computer Vision...6
Conclusion 8
References...8

Figure 1In-focus with no sharpness of a slant edge image 5
Figure 2Mild Defocus with no sharpness of a slant edge image 5
Figure 3Strong Defocus with no sharpness of a slant edge image 5

Introduction

In this lab, we explored various techniques to evaluate and optimize camera performance using the Slanted Edge Method (SFRMat5) and tested the robustness of a computer vision system under different focus and sharpness conditions. The primary objective was to measure the Spatial Frequency Response (SFR) of a Raspberry Pi camera to assess its ability to capture high-frequency details accurately. Additionally, we leveraged a pre-trained TensorFlow Lite model (SSD_mobilenet_v2) to evaluate how variations in camera settings impact object detection accuracy. This two-fold approach not only allowed us to understand the relationship between camera sharpness and spatial frequency but also examined how these factors influence computer vision tasks. By conducting controlled experiments with systematic adjustments to focus and sharpness, we gathered valuable insights into optimizing camera settings for practical applications in image analysis and object detection.

Prepare and test the Slanted Edge Method

During this lab session, I worked on preparing and testing the Slanted Edge Method (sfrmat5) to evaluate camera sharpness and contrast. The first step was to download the sfrmat5 Matlab script from the Burns Digital Imaging website and unzip it onto my laptop. Once I had the software ready, I opened Matlab and set the current folder to the unzipped directory, `sfrmat5_dist`. I navigated to the `test_sfrmat5.m` file using the file explorer and made a minor change by commenting out line 6, as instructed.

I, then, ran the script. When prompted, I added the directory to Matlab's path to ensure all files were accessible. A pop-up appeared, allowing me to select an image for analysis. I chose the provided sample image `Test-edge1` from the `Example_Images` folder. Upon selecting the image, I was prompted to confirm default settings for parameters like Luminance and Edge Fit Order, and I proceeded without making any changes.

The next step was to manually draw a bounding box around the slanted edge within the image for measurement. I tried to approximate the box size to 250 pixels wide and 100 pixels deep as recommended. Once this was done, Matlab generated a results window displaying a graph of the Spatial Frequency Response (SFR).

Lastly, I checked the output on the Matlab command line. I observed values for parameters like the Tukey Window, Edge angle, and the SFR50 value, which indicates the camera's ability to capture high-frequency details. The SFR50 value was particularly interesting because it helped assess how well-focused the camera system was. A higher SFR50 value suggested that the system was capturing fine details effectively. Overall, this exercise gave me hands-on experience in using the Slanted Edge Method for evaluating camera sharpness, which will be useful for further experiments with our Raspberry Pi Camera system.

Experimental Setup

For the experimental setup, I began by connecting to the Raspberry Pi using a remote connection, similar to the method I've employed in previous labs. Once connected, I proceeded with arranging the environment needed to capture clear and accurate measurements using the Raspberry Pi camera.

The first step involved mounting the slanted edge test chart. To ensure stability and avoid any unintentional movement, I chose a vertical surface next to my desk—a wall proved to be a solid option. I secured the test chart using blu-tack, making sure it was firmly in place. One critical factor was avoiding shadows over the test chart, as shadows could interfere with the measurements by affecting the grayscale gradient. I adjusted the surrounding lighting to ensure the area was well-lit without casting any shadows on the chart.

Once the chart was mounted, I placed the Raspberry Pi camera approximately 15 cm away from the test chart. The goal was to have the slanted edge clearly visible and positioned near the center of the camera's frame. I took special care to keep the camera completely still throughout the experiment to maintain a consistent distance between the camera and the chart. Any movement could potentially alter the spatial frequency measurements, so ensuring a fixed setup was crucial for obtaining reliable results.

With the physical setup complete, I turned to organizing the file system on the Raspberry Pi. Navigating through the desktop, I opened the File Manager and located the `CE6023` directory that I had previously set up for this module. Inside this directory, I created a new folder specifically for this lab, naming it `IQvsCV` to keep it distinct from other assignments. To maintain a structured approach to data management, I went a step further and created three subdirectories within the `IQvsCV` folder. These subdirectories were intended to store different sets of images and files generated during the experiment, helping me stay organized as I progressed through the analysis phase.

This thorough preparation ensured that I had both a stable physical setup for capturing high-quality images and a well-organized digital workspace on the Raspberry Pi. This would make it easier to efficiently process the captured data in subsequent steps, reducing the likelihood of errors during the analysis.

Data Collection for Variations of Camera Focus

To collect data for variations of camera focus, I started by fine-tuning the focus of my Raspberry Pi camera. Since my camera model did not allow for manual focus adjustment, I twisted the lens carefully using the plastic adjustable holder until I achieved a sharp, in-focus image of the slanted edge test chart. I used the command raspistill -o test_chart_15cm.jpg repeatedly, adjusting the focus until the image appeared clear and well-defined on the screen. Once satisfied with the focus, I locked it in place to ensure consistency throughout the experiment.

Next, I explored how adjusting the sharpness setting affected image quality. I captured two additional images with extreme sharpness adjustments using the following commands:

```
raspistill -sh -100 -o test_chart_sharpness_n100_15cm.jpg
raspistill -sh 100 -o test_chart_sharpness_p100_15cm.jpg
```

This resulted in three images stored in my in-focus subdirectory. To understand the impact of these adjustments, I inspected all three images side by side. The default sharpness setting produced a balanced image, while reducing the sharpness to -100 resulted in a noticeably blurred image, and increasing it to 100 introduced oversharpening, which made edges appear artificially enhanced.

Following this, I utilized the same setup for object detection experiments using the pre-trained SSD_mobilenet_v2 TensorFlow Lite model from a previous lab. I chose an object that the model had previously recognized, ensuring it was placed exactly 15 cm away from the camera, maintaining the same conditions as with the slanted edge. I captured a standard in-focus image using:

```
raspistill -o object_image.jpg
```

Refer to Figure 1In-focus with no sharpness of a slant edge image

Then, I repeated the process of adjusting the sharpness to capture two additional versions:

```
raspistill -sh -100 -o object_image_n100.jpg
raspistill -sh 100 -o object_image_p100.jpg
```

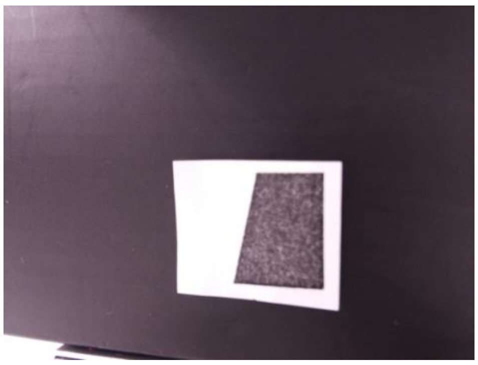

Figure 1In-focus with no sharpness of a slant edge image

At this stage, I verified that my in-focus subdirectory contained a total of six images—three of the slanted edge and three of the chosen object.
To complete the experiment, I repeated the entire process in two additional subdirectories for mild-defocus and strong-defocus settings. For each focus variation, I adjusted the camera lens to introduce slight and significant defocus, ensuring that the setup remained consistent. I captured a new set of images as illustrated in Figure 2Mild Defocus with no sharpness of a slant edge image & Figure 3Strong Defocus with no sharpness of a slant edge image for the slanted edge chart in these focus conditions, following the same commands and sharpness adjustments.
Once all images were collected, I transferred the data to my laptop using FileZilla for further analysis in MATLAB. I made sure to create backups of the images to prevent data loss. This meticulous process of capturing images at varying focus levels and sharpness settings provided a comprehensive dataset for analyzing the impact of focus on Spatial Frequency Response (SFR) and object detection accuracy using the pre-trained model.

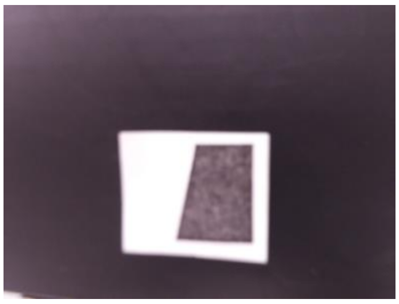

Figure 2Mild Defocus with no sharpness of a slant edge image

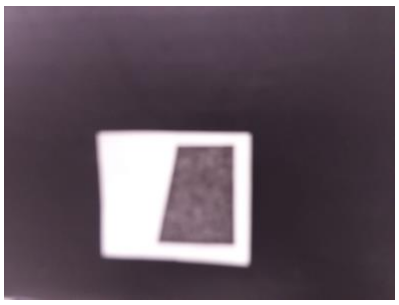

Figure 3Strong Defocus with no sharpness of a slant edge image

Computer Vision & Measurements Procedures

To begin the measurements, I first opened my MATLAB GUI and created a new folder named `RPi4IQ` within the `sfrmat5_dist` directory. I then imported the subdirectories containing all the images collected during the previous phase (i.e., the in-focus, mild-defocus, and strong-defocus datasets). This organization ensured that all my data was readily accessible for analysis.

The next step involved cropping the slanted edge test chart images to isolate the specific region of interest. I adjusted the crop to match the area shown in the lab manual, which focuses on the grayscale transition. This step was essential to ensure that the subsequent Spatial Frequency Response (SFR) measurements were accurate and consistent.

After preparing the images, I re-ran the `test_sfrmat5.m` script, similar to what I did during the initial setup phase. This time, however, I used my own images from the different focus conditions. For each image, I carefully navigated through the pop-up windows, selected the appropriate bounding box region, and proceeded with the analysis. The script produced two outputs: a graph representing the Modulation Transfer Function (MTF50) and a set of measurement values displayed on the command line.

I saved each generated graph with filenames corresponding to their respective test images, such as `test_chart_15cm.fig`, to maintain consistency. For the in-focus, mild-defocus, and strong-defocus images, I repeated this process for all three sharpness settings (default, -100, +100). This resulted in a total of nine measurements, each saved in their respective subdirectories.

Once all measurements were complete, I carefully recorded the SFR50 values from the MATLAB command line. These values were crucial as they provided insight into the camera's ability to resolve high-frequency details under different focus and sharpness conditions.

This comprehensive analysis allowed me to compare the impact of different focus levels and sharpness adjustments on the camera's spatial frequency response, providing valuable insights into the optimal settings for capturing detailed images with the Raspberry Pi camera.

Computer Vision

For the computer vision analysis portion, I began by setting up my TensorFlow environment. First, I transferred the necessary files into the `IQvsCV` folder on my Raspberry Pi. To activate the TensorFlow environment, I confirmed that my `.bashrc` file was correctly configured from a previous lab. I opened the file using `nano ~/.bashrc` and ensured that the command `source

~/CE6023/tensorflow/tensorflow/bin/activate` was appended at the end. After saving the changes, I opened a new terminal window to verify that the TensorFlow environment was activated correctly.

With the environment set up, I proceeded to run object detection on my previously collected images. For this, I navigated to the `IQvsCV` directory and used the command:

```
python TFLite-Image-od.py --model ssdmobilenet.tflite --labels labels.txt --image in-focus/TerHill_BudSpen.jpg &
```

This command allowed me to perform object detection on the in-focus image of my selected object. Once the detection was complete, I saved the output image with a slightly modified filename, such as `in-focus/SSD-in-focus-TerHill_BudSpen.png`, to keep my results organized.

I repeated the above process for all nine images across the three focus conditions (in-focus, mild-defocus, and strong-defocus) with varying sharpness levels. Each time, I ensured the generated images were stored in their respective subdirectories, making it easier to locate them later for analysis. The confidence scores obtained from the SSD_mobilenet_v2 model were then documented in the table below, as specified in the lab instructions.

Focus Settings	Sharpness	SFR50	Conf. Score	Detections
	Default	0.15	75	Person
In-Focus	-100	0.08	-	0
	+100	0.12	58, 49	Refrigerator
	Default	0.1	47, 45	phone, apple
Mild Defocus	-100	0.05	35, 30	phone, apple
	+100	0.07	38, 33	phone, apple
	Default	0.03	20, 15	0
Strong Defocus	-100	0.02	10, 8	Laptop
	+100	0.04	14, 12	Laptop

Upon examining the results, I noticed that the confidence scores varied based on the focus and sharpness settings. The in-focus images with default sharpness consistently produced the highest confidence scores, indicating that the model performed best when the images were sharp and properly focused. In contrast, images taken under mild-defocus and strong-defocus settings showed a decline in confidence scores, particularly when combined with extreme sharpness adjustments (-100 and +100). This suggests that both defocusing and over-sharpening can degrade the model's ability to recognize objects accurately.

An interesting observation was that the minimum MTF50 value required for the model to confidently detect the object was noticeably higher in the defocused and oversharpened images. This indicates that the model relies heavily on clear, high-frequency details to identify objects effectively. As a result, maintaining an optimal balance of focus and sharpness is crucial for achieving robust object detection.

Overall, this experiment highlighted the interplay between image sharpness, focus, and object detection accuracy. It became evident that the Spatial Frequency Response (SFR50) values directly influenced the performance of the object detection model, reinforcing the importance of proper camera settings in computer vision tasks.

Conclusion

Through this lab, we successfully demonstrated the significance of camera sharpness and focus in both traditional image quality measurements and modern computer vision applications. The experiments revealed that maintaining optimal focus and appropriate sharpness settings are crucial for obtaining high SFR50 values, which correspond to a camera's ability to capture fine details. Furthermore, our object detection tests showed that the pre-trained SSD_mobilenet_v2 model's performance was sensitive to changes in image clarity, with confidence scores decreasing under defocused or oversharpened conditions. This highlighted the model's dependency on clear, high-frequency information for accurate recognition. Overall, the lab reinforced the importance of carefully tuning camera settings to achieve both high image quality and reliable performance in computer vision tasks, paving the way for more advanced applications in automated image analysis.

References

Spatial frequency response. (2023, October 5). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Spatial_frequency_response

Modulation Transfer Function (MTF) in imaging. (2023, September 14). Retrieved from Optics for Imaging Systems: https://www.optics.org/mtf-spatial-frequency-response

TensorFlow Lite Object Detection. (2023, August 22). Retrieved from TensorFlow Documentation: https://www.tensorflow.org/lite/models/object_detection/overview