Technology for poorest billion GM2 AII Project
Pre-processing with optical filter

Individual report
Phupha Amornkijja (pa442), Homerton College

Introduction

The data set we obtained was derived from the AFC-330 camera, which has a cost of $11,154. Therefore, the images used to train and test the machine learning model possess exceptionally high quality. However, the device that will be constructed and utilized for the poorest billion people has a budget limit of 100 pounds. Consequently, a biconvex lens, priced at less than 10 pounds, is employed in building the device. This results in lower quality images captured by the camera. Thus, an optical filter is necessary as a pre-processing step to assess the impact of poor image quality on our machine learning model. Since the actual device has not yet been developed, we cannot guarantee that our filter accurately represents the images captured by the biconvex lens in the real device. However, by considering and manipulating six image factors, including contrast, vignette effect, lens distortion, resolution, lens flare and glare, and chromatic aberration, we can produce images that closely resemble those captured by the biconvex lens. This document will address each of these factors individually, explaining how to implement and apply them using the Python code provided in the GitHub repository. The results of predictions when implement filtered images to our machine learning model is also illustrated.

Instruction to use the optical filter python functions

The code and dataset can be accessed on GitHub. To experiment with the filter, you can execute each function separately and observe its impact on the image. Detailed instructions on how to invoke the function and specify the input image from a file directory are provided in the GitHub repository. If you wish to explore image filtering extensively, it is recommended to download the code and run it locally. Alternatively, you can upload your image to Google Drive and execute the code online. It is important to note that you have to set the file directory to obtain the input image according to where your image is stored.

Contrast

There are several factors inherent to biconvex lenses that contribute to varying contrasts in images captured by the AFC-330 camera [1]. Firstly, lens coatings play a significant role in minimizing reflections and maximizing the transmission of light through the lens, thus enhancing contrast. Secondly, the lens aperture influences image contrast, with wider apertures resulting in lower contrast due to spherical aberration. However, contrast may increase again if the aperture is too small, owing to diffraction effects. Lastly, lens quality affects contrast, as higher-quality lenses exhibit superior control over light transmission due to precise construction and manufacturing processes. Consequently, higher-quality lenses possess reduced internal reflection and can generate images with improved contrast. As biconvex lenses and the AFC-330 camera differ in these characteristics, it is necessary to apply image filtering techniques to align the contrast of AFC-330 images with those captured by biconvex lenses.

To address this, I have developed a function utilizing the Python Imaging Library (PIL) to adjust the contrast of images. The function "**change_contrast"** accepts an input image and saves the resulting image in JPG format. I utilize the JPG image format due to its advantages of requiring minimal storage space and being a commonly used format for low-resolution images. The output image's contrast can be altered by adjusting the "**contrast_factor"** parameter, which ranges from 0.0 to 2.0, with 1.0 representing the original contrast level. The images from Figure 1 to Figure 12 illustrate the effects of different contrast settings.

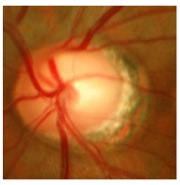


Figure1: Original image

Figure 2: Contrast = 0.0

Figure 3: Contrast = 0.1

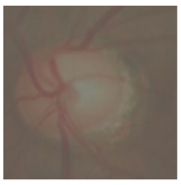


Figure 4: Contrast = 0.2

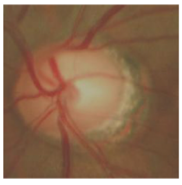


Figure 5: Contrast = 0.5

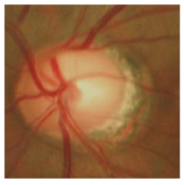


Figure 6: Contrast = 0.7

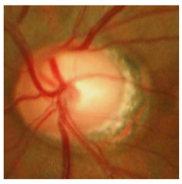


Figure 7: Contrast = 0.9

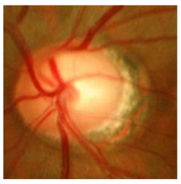


Figure 8: Contrast = 1.0

Figure 9: Contrast = 1.2

Figure 10: Contrast = 1.5

Figure 11: Contrast = 1.7

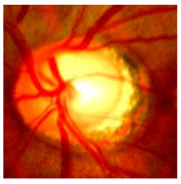


Figure 12: Contrast = 2.0

By using examples of images captured from devices, we can establish a reference point for contrast and subsequently identify the appropriate contrast level for further processing to improve our machine learning model.

Lens distortion

The presence of lens distortion is a notable characteristic affecting images captured through a biconvex lens. To address this, the "**apply_barrel_distortion**" function has been developed, enabling the conversion of input images into images exhibiting either barrel distortion or pincushion distortion, as demonstrated in the corresponding GitHub repository. When adjusting the "**distortion_factor**" parameter, values greater than 1.0 induce an intensified barrel distortion effect on the image. Conversely, values below 1.0 yield a reverse effect, resulting in a pincushion distortion. However, for the purpose of this function, our focus primarily lies on the former scenario, as biconvex lenses typically produce images with a barrel distortion effect. It is important to note that pincushion distortion commonly occurs in

zoom lenses. Please refer to the accompanying diagram depicting barrel distortion and pincushion distortion for visual reference [2].

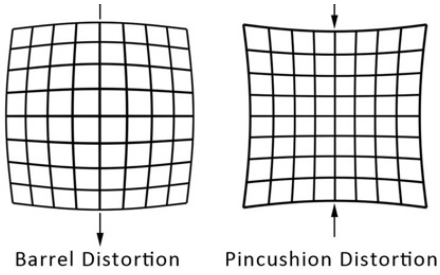


Figure 13: Illustration of Barrel Distortion and Pincushion Distortion

With research regarding aspects of biconvex lens [3], my findings indicate that a suitable range for the "**distortion_factor**" in biconvex images falls between 1.0 and 1.2. However, it is important to note that the distortion experienced in an image is influenced by several factors, including the specific characteristics of the lens, shooting conditions (such as angle and lighting), and other variables. To accurately replicate the distortion observed in actual images captured through a biconvex lens, careful calibration is necessary using real-world images. Fine-tuning the distortion parameters based on the particular biconvex lens and capturing conditions will help achieve a more accurate representation of the distortion present in the original images.

Pincushion Distortion:

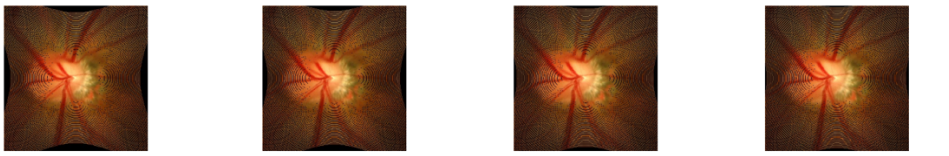


Figure14: distortion = 0.9 Figure15: distortion = 0.92 Figure16: distortion = 0.95 Figure17: distortion = 0.97

Barrel Distortion:

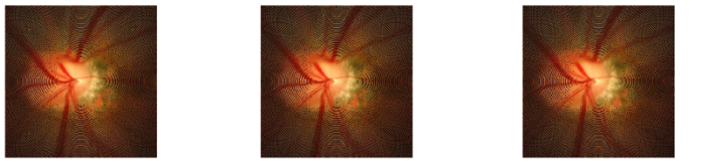


Figure18: distortion = 1.01 Figure19: distortion = 1.03 Figure20: distortion = 1.05

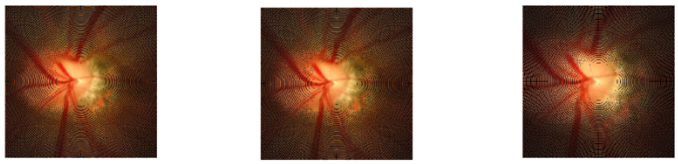


Figure21: distortion = 1.07 Figure22: distortion = 1.10 Figure23: distortion = 1.50

Vignette effect

Optical vignette caused by shading in lens barrel and intrinsic lens characteristic [4]. This phenomenon manifests as a gradual darkening or fading effect towards the corners of an image. To address this vignette effect, I have employed the **"change_vignetting"** function, utilizing the Python Imaging Library (PIL), to modify the vignette effect on the image. The **"vignette_strength"** parameter can be adjusted within the range of 0.0 to 1.0. A value of 0.0 represents the original image without any vignette effect, while a value of 1.0 signifies the maximum vignette effect.

It is worth noting that the resulting image is saved in PNG format for a specific reason. PNG format supports transparency, allowing the resulting images to possess full or partial transparency. This feature is crucial if the image requires overlaying. Therefore, saving the image in PNG format is essential. The outcome of this process is illustrated in figure 24 to figure 29.

Figure24: Original image

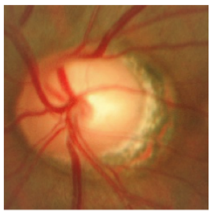


Figure25: Vignette = 0.1

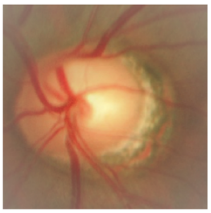


Figure26: Vignette = 0.3

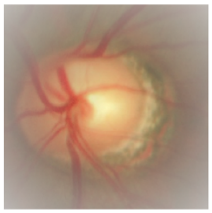


Figure27: Vignette = 0.5

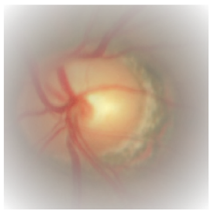


Figure28: Vignette = 0.7

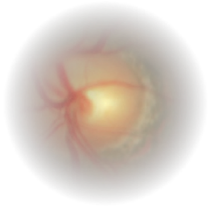


Figure29: Vignette = 1.0

By comparing these modified images with the actual images captured by the device, we can determine an appropriate value for the **"vignette_strength"** parameter. This allows us to closely resemble the image obtained through a biconvex lens.

Lens flare and glare

Lens flare is an optical phenomenon observed in images as a result of intense illumination entering the camera system, causing internal reflections and light scattering. This scattering manifests as streaks or circular patterns of light within the image. The specific characteristics of lens flare can vary depending on factors such as lens shape, quality of lens elements, and

lens coatings. Lens glare, although similar to lens flare, presents a more diffused and scattered effect when exposed to bright illumination.

To simulate the lens flare and glare effect, a function called **“apply_lens_flare_glare”** has been developed. This function generates random noise and incorporates the flare into the image at user-defined coordinates and size parameters, controlled by **“flare_center”** and **“flare_radius”.** The intensity of the flare can also be adjusted using the **“flare_intensity”** parameter. Lower values of **“flare_intensity”** result in a brighter image. Figures 30 to 35 showcase images with varying positions of the flare while maintaining a constant intensity of 0.7 and a flare radius of 10. Figures 36 to Figure 41 demonstrate images with different intensities ranging from 0.1 to 2.0. Figures 41 to Figure 46 exhibit images with varying flare radii between 10 and 100, with a fixed position at coordinate (100,150).

Varying Coordinate of flare:

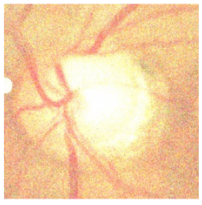


Figure30: Coordinate (0,100)

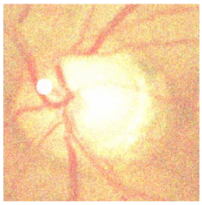


Figure31: Coordinate (50,100)

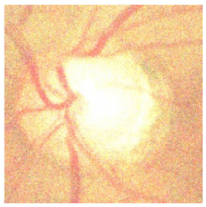


Figure32: Coordinate (100,100)

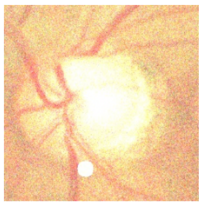


Figure33: Coordinate (100,200)

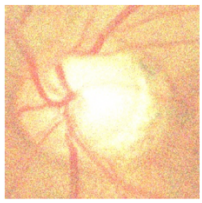


Figure34: Coordinate (200,200)

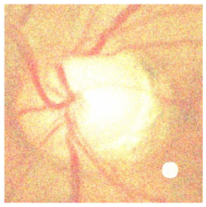


Figure35: Coordinate (200,300)

Varying Flare intensity:

Figure36: flare intensity=0.1

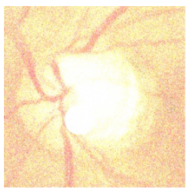


Figure37: flare intensity=0.5

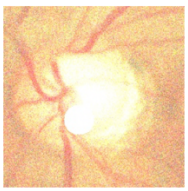


Figure38: flare intensity=0.7

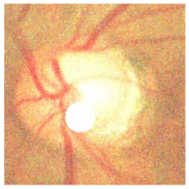


Figure39: flare intensity=0.9

Figure40: flare intensity=1.5

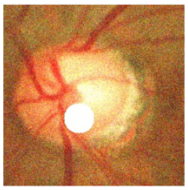


Figure41: flare intensity=2.0

Varying radius:

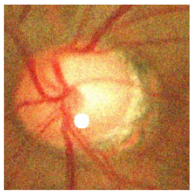


Figure41: Radius=10

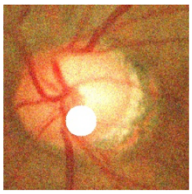


Figure42: Radius=20

Figure43: Radius=30

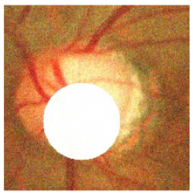


Figure44: Radius=50

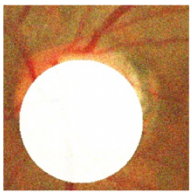


Figure45: Radius=80

Figure46: Radius=100

However, replicating lens flare and glare effects accurately proves challenging as they are contingent on the specific lighting conditions of each individual capture. The multitude of factors involved makes it impractical to implement a standardized pre-processing method. Further research is required to develop a lens flare and glare filter capable of realistically simulating image effects through a biconvex lens. It is a good idea for you to utilize the **"apply_lens_flare_glare"** function, which I have developed, as a foundation to experiment and investigate the impact of flare and glare effects on the image.

Chromatic aberration

Chromatic aberration occurs due to the variation in the focal point of light with different frequencies when passing through a lens [5]. It can be classified into two types: lateral chromatic aberration and axial chromatic aberration. This project primarily focuses on lateral chromatic aberration, which involves a horizontal shift in colours. Specifically, I have chosen to shift the red colour to the right and the blue colour to the left while keeping the green colour unchanged [6]. This intentional colour shift produces the desired chromatic aberration effect.

For the implementation, I have developed a Python function **“apply_chromatic_aberration”** utilizing the OpenCV library. The code allows for the adjustment of parameters such as **“red_shift”** and **“blue_shift”**, which determine the number of pixels by which the red and blue channels are shifted, respectively. Detailed explanations and instructions regarding the functioning of the code can be found in the comments section on GitHub. Figures 47 to 50 demonstrate the varying degrees of chromatic aberration that can be achieved by manipulating these parameters.

Figure47: Original Image

Figure48: Chromatic Shift=5

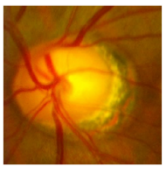


Figure49: Chromatic Shift=10

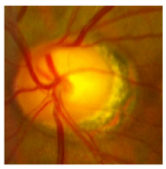


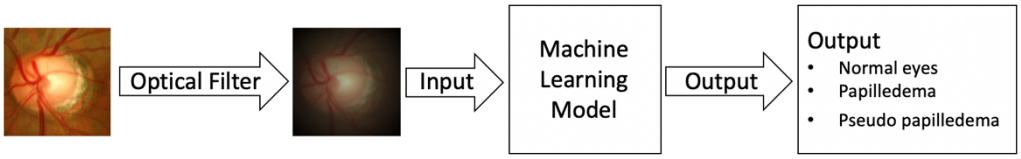
Figure50: Chromatic Shift=20

It is important to note that the manifestation of chromatic aberration is highly dependent on factors such as the dispersion of light, lens design, shooting conditions, and lens materials. Consequently, attempting to accurately replicate this phenomenon proves challenging. The presented filter serves as a starting point for further investigation and exploration in the field of optical filters.

Testing and results:

By combining all the factors mentioned above, an optical filter can be created and applied to an image, resulting in an image that closely resembles one captured through a biconvex lens. I have employed a combination of these factors to provide an illustration of how they can be merged and explored within our machine learning model.

The investigation involves applying the optical filter to a test image and assessing whether the model can accurately predict the image's classification, distinguishing between Normal eyes, Papilledema, and Pseudo papilledema. The results and the visual representation of my workflow are presented in Figure 20.



To conduct the machine learning test, you have the option to download the model of your choice from GitHub onto your computer. The testing code can also be downloaded and executed locally. It is important to note that users should pay attention to the file directory, as it may vary across different computers when running the code locally. Running the code locally is advisable when working with images so that you can experiment with your choice of optical filter. I would like to acknowledge and express gratitude to Roshen for providing a helpful suggestion on developing the code for prediction testing that is compatible to run locally.

The initial optical filter that I examined focuses solely on two factors: contrast and vignette effect. These two factors were chosen due to their relative ease of imitation and limited variability. The code for this filter has been developed as a function named **"apply_image_effects2"**, which is available on GitHub. The code is displayed with annotations

in the repository, providing explanatory comments. The outcomes of the test conducted using this optical filter are presented in Table 1.

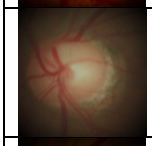
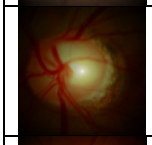
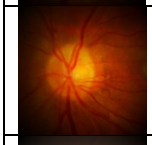
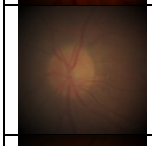
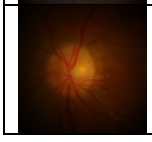
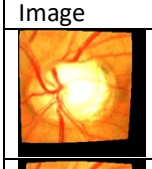
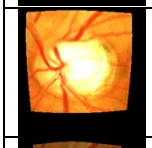
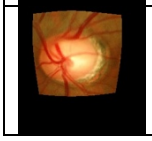
Image	Parameters	Prediction from our model	Actual label
	Contrast = 2 Vignette = 1.1	Normal	Normal
	Contrast = 0.5 Vignette = 1.5	Normal	Normal
	Contrast = 1.5 Vignette = 0.5	Normal	Normal
	Contrast = 2 Vignette = 1.1	Papilledema	Papilledema
	Contrast = 0.5 Vignette = 1.5	Papilledema	Papilledema
	Contrast = 1.5 Vignette = 0.5	Papilledema	Papilledema

Table 1: Results from pre-processing with contrast and vignette effect

The subsequent optical filter that I evaluated incorporates lens distortion, flare and glare effects, as well as chromatic aberration. These factors were chosen to simulate more realistic image distortions. The code for this filter is implemented as a function named **"apply_image_effects3**", and it is available on GitHub. The code is presented with explanatory comments. The outcomes of the test conducted using this optical filter are displayed in Table 2.

Image	Parameters	Prediction from our model	Actual label
	Distortion = 0.001 Glare centre = (50,50) Glare radius = 10 Glare intensity = 0.5 Red_shift = 0 Blue_shift = 1	Normal	Normal
	Distortion = 0.002 Glare centre = (100,50) Glare radius = 15 Glare intensity = 0.5 Red_shift = 0 Blue_shift = 1	Normal	Normal
	Distortion = 0.005 Glare centre = (100,50) Glare radius = 10 Glare intensity = 1.0 Red_shift = 1 Blue_shift = 3	Normal	Normal

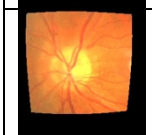
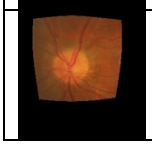
	Distortion = 0.001 Glare centre = (50,50) Glare radius = 10 Glare intensity = 0.5 Red_shift = 0 Blue_shift = 1	Papilledema	Papilledema
	Distortion = 0.002 Glare centre = (100,50) Glare radius = 15 Glare intensity = 0.5 Red_shift = 0 Blue_shift = 1	Papilledema	Papilledema
	Distortion = 0.005 Glare centre = (100,50) Glare radius = 10 Glare intensity = 1.0 Red_shift = 1 Blue_shift = 3	Papilledema	Papilledema

Table 2: Results from pre-processing with lens distortion, flare and glare effects and chromatic aberration

Conclusion

In this project, I examined six factors influencing the optical filter. To address these factors, I developed five Python functions focused on contrast, vignette effect, lens distortion, lens flare and glare, and chromatic aberration. Additionally, I explored the combined effects of these factors on our machine learning predictions. By utilizing these Python functions to create optical filters and developing a universal testing function, I evaluated the results of our model using two optical filter images. Despite significant optical modifications, our model consistently achieved accurate image type predictions.

The strength of this project is that the construction of each filter within a function facilitates easy exploration and future utilization of optical filters. Furthermore, the implementation of the testing function signifies the completion of the entire process involved in building optical filters. While numerous pre-processing techniques can be employed to manipulate images as described above, a significant challenge lies in the absence of a physical device capable of producing an output image resembling that of a human eye. Consequently, a direct comparison between our pre-processed images and actual eye-captured images is unattainable. Hence, a much deeper investigation of how optical filter effects our machine learning model can be achieved from the availability of an actual device or a representative image captured using a biconvex lens.

A potential future work to this project involves the inclusion of additional factors for optical filters, such as noise reduction, white balance, and depth of field. These factors hold substantial importance in the overall effectiveness of optical filters. However, it is worth noting that the impact of these additional developments may not be prominently discernible in the absence of actual images captured through a biconvex lens using an authentic device. Up till now, the pre-processed images were exclusively utilized for testing purposes in the trained model which uses high-quality image data set taken by AFC-330 for training. Potential future research could involve training the model using these pre-processed images and subsequently evaluating its performance to determine if it yields higher confidence levels than the machine learning model trained from high-quality data set.

Reference:

[1] John, M., et al. “Learners’ Conceptual Understanding about Image Formation by a Convex Lens: How and to What Extent Can It Be Improved?” International Journal for Cross-Disciplinary Subjects in Education, vol. 9, no. 2, 30 June 2018.

[2] “Look out for Lens Distortion.” Learning with Experts, www.learningwithexperts.com/photography/blog/look-out-for-lens-distortion.

[3] Podoleanu, Adrian & Charalambous, Ismini & Plesea, Lucian & Dogariu, Aristide & Rosen, Richard. (2004). Correction of distortions in optical coherence tomography imaging of the eye. Physics in medicine and biology.

[4] Red.com, 2023.

[5] “Chromatic Aberration | Understanding Chromatic Aberration | Nikon.” Www.nikonusa.com, www.nikonusa.com/en/learn-and-explore/a/products-and-innovation/chromatic-aberration.

[6] “Simulate Lens Flare and Chromatic Aberration Using Python.” Stack Overflow, stackoverflow.com/questions/50736708/simulate-lens-flare-and-chromatic-aberration-using-python.