深 圳 大 学 实 验 报 告

课程名称：数字图像处理

实验项目名称：**Exp1 Basic Operations and Algebraic Operators for Digital Images**

学院：电子与信息工程学院

专业：电子信息工程

指导教师：李斌

报告人：廖祖颐 学号：**2022110131** 班级：文华班

实验时间：**2024** 年 **3** 月 **5** 日、**12** 日、**19** 日、**26** 日

实验报告提交时间：**2024** 年 **3** 月 **27** 日

教务部制

实验目的（Aim of Experiment）：

(1) Establish a Python environment and install some digital image processing libraries such as scikit-image, OpenCV, PIL, and matplotlib.
(2) Learn how to load, display, and save images.
(3) Be familiar with some basic image processing operations such as adding noise, image type conversion, image file format conversion, etc.
(4) Learn how to perform algebraic operations on digital images.

实验内容与要求（Experiment Steps and Requirements）：

(1) **Load, Save and Display Images.** (a) Load a PNG image with Scikit-Image. (Tips: io.imread). (b) Convert it to PIL image format. (Tips: Image.fromarray). (c) Display this image with Matplotlib. (d) Convert PIL image format to OpenCV image format, save as a JPEG image with a quality factor of 90. (Tips: cv2.imwrite).
(2) **Display Three Individual Color Components of RGB Image.** (a) Load an image with OpenCV. (b) Display the R, G, and B color component of image, respectively. Display them in the same figure with sub-figures. Add the corresponding title to the sub-figure. (c) Answer the question: What are the differences in R, G, and B color components?
(3) **Convert Color Image to Grayscale.** (a) Display the original Lena image and the grayscale images obtained by three grayscaling methods in the same figure. Add the corresponding title. a1) Maximum of the three components; a2) The average of the three components; a3) gray=0.30r + 0.59g + 0.11*b. (b) Answer the question: What are their differences?
(4) **Image Cropping.** (a) Load a RGB image. (b) Select the 128x128 central region of the image. (c) Display the full image and its central part. (d) Save the central part as an image file in the same format as the full image.
(5) **Adding Noise to Image.** (a) Load an RGB image in with Scikit-Image. (b) Add Gaussian additive noise, salt noise, pepper noise, salt and pepper noise, or speckle noise to it. The parameters can be chosen by yourself. (Tips: You may use random_noise in the util module of Scikit-Image). (c) Display these six images in the same figure and add the corresponding title.
(6) **Image Denoising by Averaging.** (a) Load an RGB image. (b) Add Gaussian noise with a mean value of 0 and a variance of 0.1 to it. (c) Display and compare the images before and after adding noise. (d) Use the for loop to add 3, 30, and 300 images with random Gaussian noise and find their average value, respectively.
(7) **Image Algebraic Operations.** (a) Download two pictures by yourself, and load these two pictures with OpenCV. (b) Perform algebraic operations of addition and subtraction (Tips: Pay attention to the size and type of the image during the calculation. If they are different, the larger image should be cropped or scaled). (c) Display the images before and after processing.
(8) **Text adding.** (a) Load the Lena image with OpenCV. (b) Employ a red rectangle to mark the 64x64 rectangle in the middle of image. (Tips: You may use cv2.rectangle) (c) Adding some black text on it. (Tips: You may usecv2.putText). An example of the

generated result is shown in the figure below.

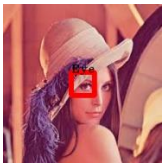


Figure 1. The result of adding content to Lean image

(9) **Adding image mask.** (a) Load the Lena image with OpenCV. (b) Adding a circular mask on the Lena image. (Tips: You may usenumpy.ogrid). An example of the generated result is shown in the figure below.

Figure 2. The result of adding mask to Lean image

(10) **Capture Images by Laptop Camera with OpenCV. (Bonus Practice).** (a) Read the video stream from the laptop camera. (b) Implement Time-lapse photography.(It can be used to record the whole process of the evaporation of water droplets in the cup. Tips: You may usetime.sleep). (c) Generate corresponding MP4 video or GIF image.

实验代码及数据结果（Experiment Codes and Results）：

1. Load, Save and Display Images

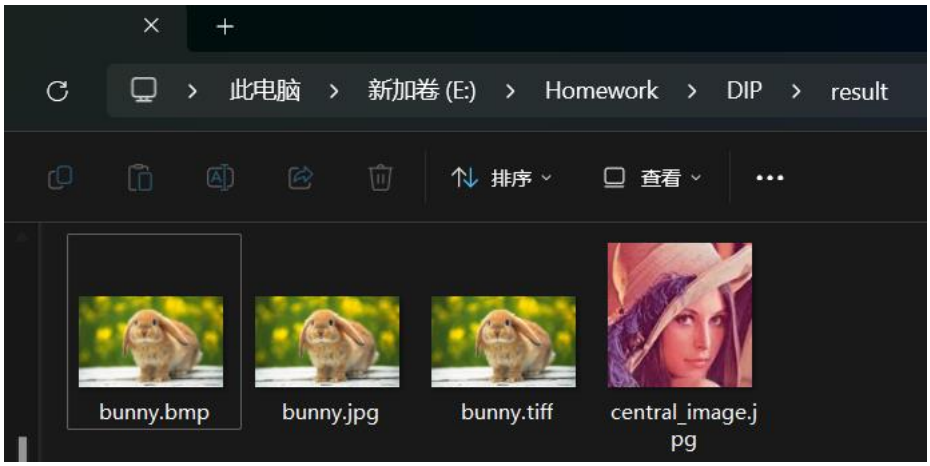
code

```
from skimage import io
from PIL import Image
import matplotlib.pyplot as plt
import cv2
import numpy as np

# (a) Load a PNG image
bunny = io.imread(r"H:\大二下\数图\Exp1\images\bunny.png")
print(bunny.shape)

# (b)  Convert it to PIL image format
bunny = Image.fromarray(bunny)

# (c) Display this image
plt.imshow(bunny)
plt.axis('off')
plt.show()

# (d) Convert image format
bunny = np.array(bunny)
bunny = cv2.cvtColor(bunny, cv2.COLOR_RGB2BGR)# Converting an image from RGB color space to BGR color space.
cv2.imwrite(r'E:\Homework\DIP\result\bunny.jpg', bunny, [cv2.IMWRITE_JPEG_QUALITY, 90])

# (e) Load the BMP format
cv2.imwrite(r'E:\Homework\DIP\result\bunny.bmp',bunny)

# (e) Load the TIFF format
cv2.imwrite(r'E:\Homework\DIP\result\bunny.tiff',bunny)
```

result

(362, 580, 4)

True

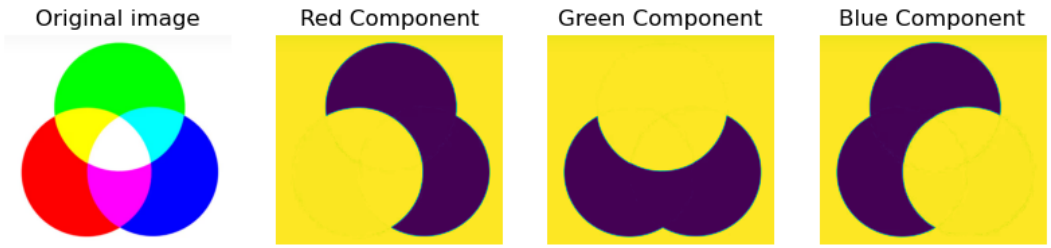


2. Display Three Individual Color Components of RGB Image

code

```python
# (a) Load an image with OpenCV
image = cv2.imread(r"E:\Homework\DIP\image\three primary colours.jpg")
# Convert BGR to RGB
original_image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

# (b) Separate the R, G, and B color components of the image
R, G, B = cv2.split(original_image)

# Display them in the same figure with sub-figures
plt.figure(figsize=(10,4))

# Original image
plt.subplot(1, 4, 1)
plt.imshow(original_image)
plt.title('Original image')
plt.axis('off')

# Red Component image
plt.subplot(1, 4, 2)
plt.imshow(R)
plt.title('Red Component')
plt.axis('off')

# Green Component image
plt.subplot(1, 4, 3)
plt.imshow(G)
plt.title('Green Component')
plt.axis('off')

# Blue Component image
plt.subplot(1, 4, 4)
plt.imshow(B)
plt.title('Blue Component')
plt.axis('off')

plt.show()

# Answer question
# RGB image is consist of red channel, blue channel and green channel.
# RGB image contains more color visually due to colour mixing . By cont
```

result



Due to I didn't convert the image into gray-image, so the brighter place mean the matching component. The white is the RGB(255,255,255), so each component contain the white background.

3. Convert Color Image to Grayscal

code

```python
# Load the image
image = cv2.imread(r"E:\Homework\DIP\image\lena.jpg")
img = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

# Separate the R, G, and B color components of the image
R, G, B = cv2.split(img)

# 1) Maximum of the three components
gray_max_image = np.maximum(np.maximum(R, G), B)

# 2) The average of the three components
gray_avg_image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 3) Weighted grey value gray=0.30r + 0.59g + 0.11*b
gray_weighted_image = (0.30 * R + 0.59 * G + 0.11 * B).astype(np.uint8)


# Display
plt.figure(figsize=(12, 6))

# Original image
plt.subplot(1, 4, 1)
plt.imshow(img, cmap = 'gray')
plt.title('Original image')
plt.axis('off')

# Max component image
plt.subplot(1, 4, 2)
plt.imshow(gray_max_image, cmap = 'gray')
plt.title('Max component image')
plt.axis('off')

# Average component image
plt.subplot(1, 4, 3)
plt.imshow(gray_avg_image, cmap = 'gray')
plt.title('Average component image')
plt.axis('off')

# Weighted sum image
plt.subplot(1, 4, 4)
plt.imshow(gray_weighted_image, cmap = 'gray')
plt.title('Weighted sum image')
plt.axis('off')

# The question
# 1) The resulting greyscale image tends to be brighter in terms of bright
# This can result in some detail in the image being lost in the brighter a
# 2) The generated greyscale image is moderate in brightness and retains t
# 3) The resulting greyscale image is visually more balanced and natural,
```

result

4. Image Cropping

code

```python
# (a) Load an RGB image
image = cv2.imread(r"E:\Homework\DIP\image\lena.jpg")
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Get the image size
height, width, _ = image.shape

# The wanted image size
size_height = 128
size_width = 128

# Select the starting point
x_center = width // 2
y_center = height // 2
x_start = x_center - size_height // 2
y_start = y_center - size_width // 2

# (b) Select the 128x128 central region of the image.
central_image = image[y_start:y_start + size_height, x_start:x_start + size_width]

# Display
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.title("Full Image")
plt.imshow(image)
plt.axis('off')

plt.subplot(1, 2, 2)
plt.title("Central Image")
plt.imshow(central_image)
plt.axis('off')

plt.show()

# (d) Save the image
cv2.imwrite(r"E:\Homework\DIP\result\central_image.jpg", cv2.cvtColor(central_image, cv2.COLOR_RGB2BGR))
```

result

5. Adding Noise to Image

code

```python
from skimage import util

# (a) Load an RGB image
image = io.imread("E:\Homework\DIP\image\lena.jpg")

# (b) Add different types of noise to the image
gaussian_noise = util.random_noise(image, mode='gaussian')
salt_noise = util.random_noise(image, mode='salt')
pepper_noise = util.random_noise(image, mode='pepper')
salt_pepper_noise = util.random_noise(image, mode='s&p')
speckle_noise = util.random_noise(image, mode='speckle')

# (c) Display all the images
plt.figure(figsize=(15, 10))

# Original image
plt.subplot(2, 3, 1)
plt.imshow(image)
plt.title("Original")
plt.axis('off')

# Gaussian Noise
plt.subplot(2, 3, 2)
plt.imshow(gaussian_noise)
plt.title("Gaussian Noise")
plt.axis('off')

# Salt Noise
plt.subplot(2, 3, 3)
plt.imshow(salt_noise)
plt.title("Salt Noise")
plt.axis('off')

# Pepper Noise
plt.subplot(2, 3, 4)
plt.imshow(pepper_noise)
plt.title("Pepper Noise")
plt.axis('off')

# Salt & Pepper Noise
plt.subplot(2, 3, 5)
plt.imshow(salt_pepper_noise)
plt.title("Salt & Pepper Noise")
plt.axis('off')

# Speckle Noise
plt.subplot(2, 3, 6)
plt.imshow(speckle_noise)
plt.title("Speckle Noise")
plt.axis('off')

plt.show()
```

result

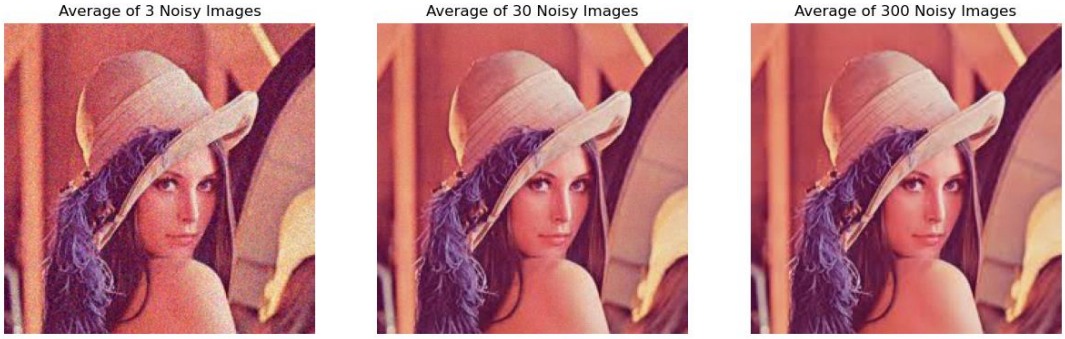
6. Image Denoising by Averaging

code

```
# (a) Load an RGB image
image = io.imread("E:\Homework\DIP\image\lena.jpg")
image = np.array(image)

# (b) Add Gaussian noise
gaussian_image = util.random_noise(image, mode = 'gaussian', mean = 0, var = 0.1, clip = True )

# Display the image
plt.figure(figsize=(10, 5))

# Display the 'before' image
plt.subplot(1, 2, 1)
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')

# Display the 'after' image
plt.subplot(1, 2, 2)
plt.imshow(gaussian_image)
plt.title('Gaussian Noise Image')
plt.axis('off')

plt.show()

# Function to add noise and average images
def average_noisy_image(image, num_images):
    gaussian_noise = np.zeros(image.shape, dtype = np.float64)
    for _ in range(num_images):
        gaussian_noise += util.random_noise(image, mode = 'gaussian')
    gaussian_noise = gaussian_noise * 255
    gaussian_noise = (gaussian_noise / num_images).astype(np.uint8)
    return gaussian_noise
```

```
noise_gaussian_image_3 = average_noisy_image(image, 3)
noise_gaussian_image_30 = average_noisy_image(image, 30)
noise_gaussian_image_300 = average_noisy_image(image, 300)

# Display the image
plt.figure(figsize=(15, 10))

# Average of 3 Noisy Images
plt.subplot(1, 3, 1)
plt.imshow(noise_gaussian_image_3)
plt.title('Average of 3 Noisy Images')
plt.axis('off')

# Average of 30 Noisy Images
plt.subplot(1, 3, 2)
plt.imshow(noise_gaussian_image_30)
plt.title('Average of 30 Noisy Images')
plt.axis('off')

# Average of 300 Noisy Images
plt.subplot(1, 3, 3)
plt.imshow(noise_gaussian_image_300)
plt.title('Average of 300 Noisy Images')
plt.axis('off')
plt.show()

# Image averaging can reduce noise
```

result

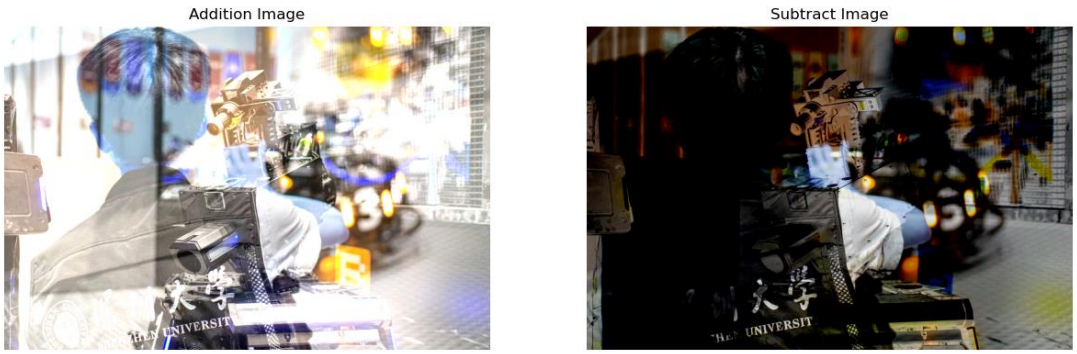


7. Image Algebraic Operations

code

```
# (a) Load two pictures
image_1 = cv2.imread(r"E:\Homework\DIP\image\9713ccc42d067a56773324c4d30fc7d.jpg")
image_2 = cv2.imread(r"E:\Homework\DIP\image\55337effbd9e1345646ea755da390f5.jpg")

# (b) Algebraic operations of addition
add_image = cv2.add(src1 = image_1, src2 = image_2)

# (b) Algebraic operations of subtraction
sub_image = cv2.subtract(src1 = image_1, src2 = image_2)

plt.figure(figsize=(15, 10))

plt.subplot(1, 2, 1)
plt.imshow(add_image)
plt.title('Addition Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(sub_image)
plt.title('Subtract Image')
plt.axis('off')

plt.show()
```

result

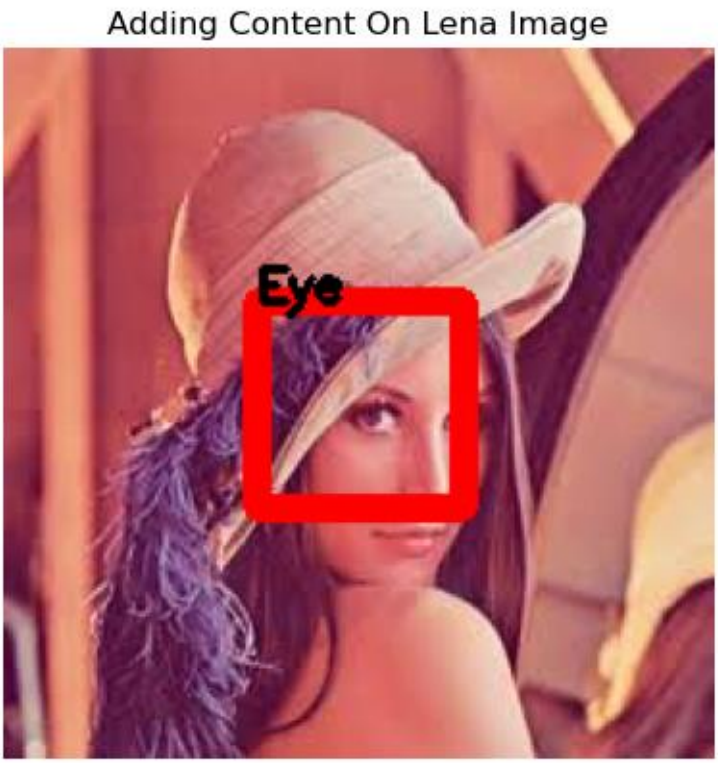


8. Text adding

code

```python
# (a) Load the Lena image
image = cv2.imread(r"E:\Homework\DIP\image\lena.jpg")
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# The wanted image size
size_height = 64
size_width = 64

# Get image size
height, width, _ = image.shape

x_center = width // 2
y_center = height // 2
x_start = x_center - size_height // 2
y_start = y_center - size_width // 2
# (b) Employ a red rectangle
cv2.rectangle(image, (x_start, y_start), (x_start + size_width, y_start + size_height), (255, 0, 0), 8)
# (c) Adding black text
cv2.putText(image, 'Eye', (x_start, y_start), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 2)

# Display the image
plt.imshow(image)
plt.title('Adding Content On Lena Image')
plt.axis('off')
plt.show()
```

result



9. Adding image mask

code

```python
# (a) Load the Lena image
image = cv2.imread(r"E:\Homework\DIP\image\lena.jpg")
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# (b) Adding a circular mask on the Lena image
height, width, _ = image.shape
x, y = np.ogrid[0:height, 0:width]
mask = (x - height / 2) ** 2 + (y - width / 2) **2 > height * width / 4
image[mask, :] = 0

# Display the image
plt.imshow(image)
plt.title('Circular Mask On Lena Image')
plt.axis('off')
plt.show()
```

result

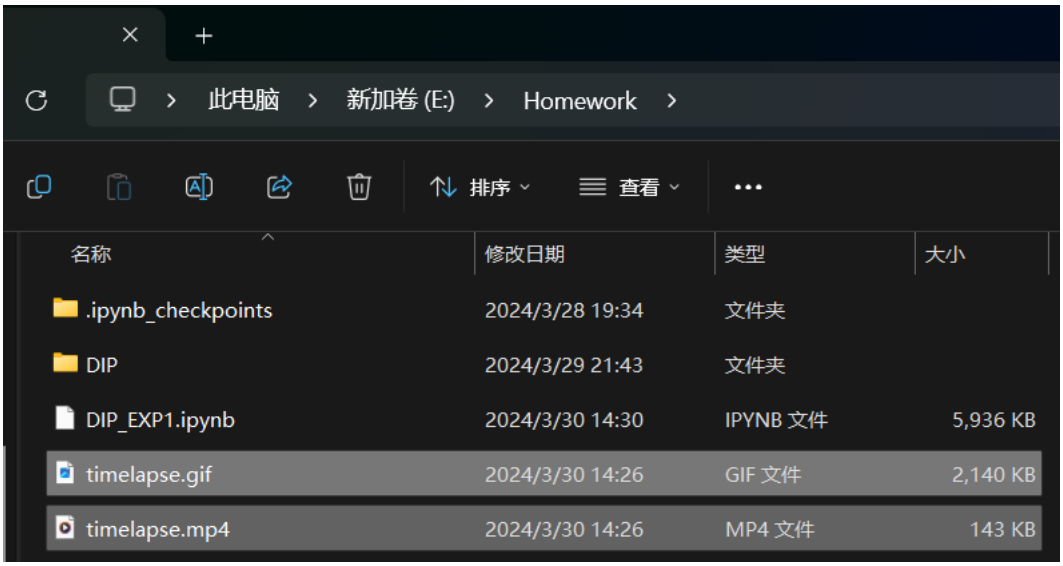
10. Capture Images by Laptop Camera with OpenCV. (Bonus Practice)

code

```python
import cv2
import time
import imageio

def capture_timelapse(interval, duration, output_video, output_gif):
    """
    Capture time-lapse images from the webcam and save as video and GIF.

    :param interval: Time in seconds between frames.
    :param duration: Total duration of the timelapse in seconds.
    :param output_video: Filename for the output video.
    :param output_gif: Filename for the output GIF.
    """
    cap = cv2.VideoCapture(0)  # Initialize the camera

    if not cap.isOpened():
        print("Could not open camera.")
        return

    frames = []
    start_time = time.time()

    while True:
        ret, frame = cap.read()  # Capture frame-by-frame

        if not ret:
            print("Can't receive frame")
            break

        frames.append(frame)  # Store the frame for later processing
        time.sleep(interval)  # Wait for the specified interval

        # Show the frame
        cv2.imshow('Timelapse Frame', frame)

        if cv2.waitKey(1) == ord('q') or (time.time() - start_time) > duration:
            break

    cap.release()  # Release the camera
    cv2.destroyAllWindows()  # Close all OpenCV windows

    # Save the frames to a video file
    height, width, layers = frames[0].shape
    video = cv2.VideoWriter(output_video, cv2.VideoWriter_fourcc(*'mp4v'), 1.0 / interval, (width, height))

    for frame in frames:
        video.write(frame)

    video.release()

    # Save the frames to a GIF file
    imageio.mimsave(output_gif, frames, 'GIF', duration=interval)

# Parameters
interval = 2  # seconds
duration = 20  # seconds
output_video = 'timelapse.mp4'
output_gif = 'timelapse.gif'

capture_timelapse(interval, duration, output_video, output_gif)
```

result


媒体播放器
0:00:08
0:00:12
timelapse


此电脑 › 新加卷 (E:) › Homework ›
排序
查看
名称
修改日期
类型
大小
.ipynb_checkpoints 2024/3/28 19:34 文件夹
DIP 2024/3/29 21:43 文件夹
DIP_EXP1.ipynb 2024/3/30 14:30 IPYNB 文件 5,936 KB
timelapse.gif 2024/3/30 14:26 GIF 文件 2,140 KB
timelapse.mp4 2024/3/30 14:26 MP4 文件 143 KB

深圳大学学生实验报告用纸

实验分析与结论（Analysis and Conclusion）：

Experiment Analysis and conclusion:

- **Load, Save and Display Images**

1.When we load a image using io or PIL, the sequence of three components is RGB. Instead, when we load a image using cv2, the sequence of three component is BGR.

2.Then the image loaded by io is capable of numerical operation directly since its format is array.

3.Cv2 can only read the English file path, others can read the Chinese file path.

4.All the three image format(PIL/cv2/skimage) can convert to each other.

5.We can save any reasonable image format by cv2.imwrite('imgage_name.file_format', image). But we should carefully notice the channels because when we dealing with the image, we always deal the RGB channel, at the final saving image part, we should convert the image component to BGR in order to getting a normal image.

- **Display Three Individual Color Components of RGB Images**

1. As said in the part one, the image whose format is cv2 has a different color components sequence(BRG), so when these images are split, the output of the split function is B,G,R in order.

2.The brighter one mean the channel component

3.What are the differences between the RGB image and the R/G/B components?

(1) RGB image is consist of red channel, blue channel and green channel. It's just what we see every day.

(2) RGB image contains more color visually due to colour mixing . By contract, R/G/B

components emphasize its corresponding channel color.

- **Convert Color Image to Grayscale**

1. In the experiment of getting grayscale, we actually do operation for pixels.

2.We can use different operation to obtain different grayscale.

3.What are their differences?

1) The resulting greyscale image tends to be brighter in terms of brightness because it tends to select the brightest parts of the colour channels. This can result in some detail in the image being lost in the brighter areas, especially if there is a large difference in the colour components in the original image.

2) The generated greyscale image is moderate in brightness and retains the visual information of the original image better. Since all colour channels are given the same weight, the differences in the sensitivity of the human eye to different colours may be ignored to some extent, and the details of certain colours are not prominent enough.

3) The resulting greyscale image is visually more balanced and natural, and better reflects the human eye's true perception of different colours.

- **Image Cropping**

1.In the experiment of image cropping, we actually find out the position of boundary pixels for the region we want. Once we decide the four corners position of the image border, image cropping is done.

2.With the image cropping operation, we can cut out region whatever we want in the image.

- **Adding Noise to Image**

1. Different noise type influence the quality of the image to varying degrees. Adding speckle noise is less obvious than adding others noise.

2. When two noise are added at the same time in one image, it equivalent to add the two noise respectively. And the effect of two approaches is same.(It feels as if it's just a normal superposition of two noise effects.)

- **Image Denoising by Averaging**

1. When we add more images with random Gaussian noise and divide its add times, we will receive a more clear image. Since it decreases the variance of the calculative noise and the mean of gaussian noise is zero, consequencely, the noise will gradually eliminate as it increases.

2. Image averaging can reduce noise

- **Image Algebraic Operations**

1. If we carry out image algebraic operations between two images, their size and type should be fit.

2. When two cv2 format images get ready to add, we should use cv2.add() function to achieve instead of adding their pixel value. When we use arithmetic method to achieve image algebraic operations, the pixels value format should be converted to float(img_as_float).

3. The adding and subtract method is adding some effect like movie image. They can both do the task like the ‘Adding image mask’. Just need to generate the diagram what we need.

4. When the two bright image are added, the result will be more bright; When the bright region are subtracted, the result will be dark.

5. I also try the image multiplication and division. This two way always make the Pixel values above 255 or below 0.

- **Text adding**

Similar to image cropping, we actually find the position of the four corners of the border. And the text is also put on target pixel position.

- **Adding image mask.**

The shape of the mask we add is a circular mask. First of all, we set up a matrix ’mask’ which has the same shape as original image in convince to record the position information. Then take the central point of the image as the center of the circle and find out the pixels outside this circle. Assign True value for these outside points in matrix ‘mask’. Finally let these pixels value to zero. Output the image and we will find the region outside the circle is black duo to its pixels value is zero.

- **Capture Images by Laptop Camera with OpenCV. (Bonus Practice).**

Actually, the principle of generating video, GIF image or cv2 window is grouping multiple frames together so it seems like a continuous process. Control the time interval of frame capture by time.sleep(interval) to achieve the time delay effect.

Experiment summary:

In this experiment, I learn the basic operation for image process. At first, we should use specific function to load a image and convert its format if necessary. The we can achieve some basic operation for image such as convert color image to grayscale, image cropping, adding noise to image, image denoising by averaging, image algebraic operations, text adding or adding image mask etc. Each basic operation can achieved by calling corresponding function or execute corresponding pixel-wise value operation. Finally, we can save the processed image in any format by using specific function. For the bonus part, I learn how to capture real-time image through built-in camera in my laptop. And use the frame to make video or GIF image. Also, I roughly gain the principle of time-lapse photography. Using this skill, I can record the whole process which is relative slow.

Overall, these experimental steps provide a comprehensive introduction to the core concepts of image processing and deepen understanding through hands-on practice, demonstrating the ability to utilise the major image processing libraries for image analysis and editing in a Python environment.

指导教师批阅意见：

成绩评定：

实验态度 10 分	实验步骤及代码 40 分	实验数据与结果 40 分	实验分析与结论 10 分

指导教师签字：李斌

2024 年 3 月 30 日

备注：