

系统开发工具基础课程实验报告

姓名：张誉馨

2024 年 9 月 16 日

目录

1	练习内容和结果	1
1.1	python基础	1
1.2	python视觉应用	2
1.3	命令行环境	6
2	解题感悟	10
3	GitHub链接	11

1 练习内容和结果

1.1 python基础

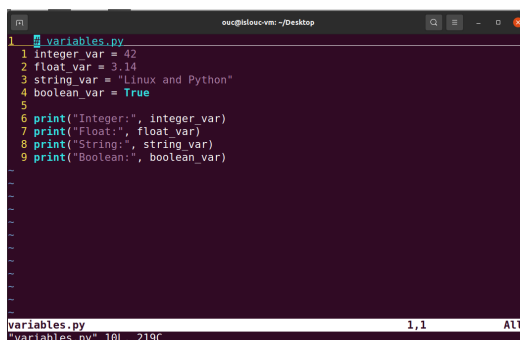
1.hello world

001helloworld.py里的内容是: `print("Hello, Python!")`

```
ouc@islouc-vm:~/vim/pack/vendor/start$ cd ~/Desktop/class/python_codes
ouc@islouc-vm:~/Desktop/class/python_codes$ touch 001helloworld.py
ouc@islouc-vm:~/Desktop/class/python_codes$ vim 001helloworld.py
Error detected while processing /home/ouc/.vimrc:
line 15:
E492: Not an editor command: 语法高亮
line 48:
E492: Not an editor command: 然后把它保存到 ~/.vimrc! Comments in Vimscript start with a `
Press ENTER or type command to continue
ouc@islouc-vm:~/Desktop/class/python_codes$ python3 001helloworld.py
Hello world
```

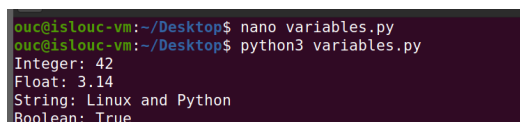
图 1: hello world

2.创建一个 Python 脚本进行变量定义和操作



```
ouc@islouc-vm:~/Desktop$ nano variables.py
1 integer_var = 42
2 float_var = 3.14
3 string_var = "Linux and Python"
4 boolean_var = True
5
6 print("Integer:", integer_var)
7 print("Float:", float_var)
8 print("String:", string_var)
9 print("Boolean:", boolean_var)
```

图 2: 创建python脚本variables.py进行变量定义和操作



```
ouc@islouc-vm:~/Desktop$ nano variables.py
ouc@islouc-vm:~/Desktop$ python3 variables.py
Integer: 42
Float: 3.14
String: Linux and Python
Boolean: True
```

图 3: 运行脚本variables.py

3.掌握 Python 中的控制结构和函数定义。

4.学习如何定义和调用函数。

5.菱形

```
1 # control_structures.py
2
3 number = 10
4
5 if number > 5:
6     print("Number is greater than 5")
7 else:
8     print("Number is 5 or less")
9
10 for i in range(3):
11     print("For Loop iteration:", i)
12
13 count = 0
14 while count < 3:
15     print("While Loop iteration:", count)
16     count += 1
17
18 def greet(name):
19     return f"Hello, {name}!"
20
21 print(greet("World"))
```

图 4: 创建python脚本control_structures.py测试控制结构

```
huc@isluc-vm: ~/Desktop$ name control_structures.py
huc@isluc-vm: ~/Desktop$ vim control_structures.py
Error detected while processing /home/ouc/.vimrc:
line 15:
E492: Not an editor command: 语法高亮
line 48:
E492: Not an editor command: 然后把它保存到 ~/.vimrc! Comments in Vimscript start with a
".
Press ENTER or type command to continue
huc@isluc-vm: ~/Desktop$ python3 control_structures.py
Number is greater than 5
For Loop iteration: 0
For Loop iteration: 1
For Loop iteration: 2
While Loop iteration: 0
While Loop iteration: 1
While Loop iteration: 2
Hello, World!
```

图 5: 执行control_structures.py脚本

6. 下面是一个示例 Python脚本loop.py，它演示了如何使用不同类型的循环（for 和 while）来处理基本的循环任务。这个脚本将打印从 1 到 5 的数字，并在 while 循环中计算数字的平方。

7.下面是一个简单的 Python 脚本，用于判断一个给定的年份是否为闰年

1.2 python视觉应用

1.PIL: Python 图像处理类库。

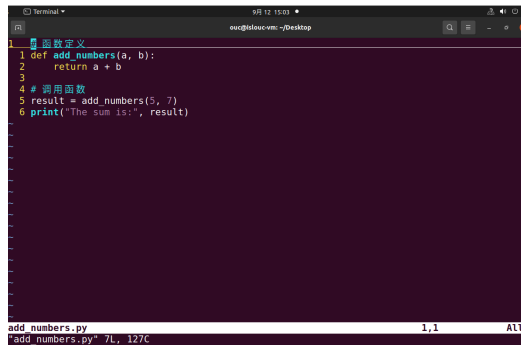
写个简单的Python程序，完成以下功能： a)打开一幅图片（如自己的照片）

b)将图片大小修改成640*480

c)将修改大小后的图像转成黑白图像

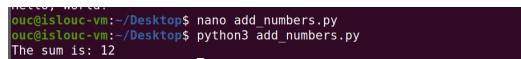
d)将黑白图像存成gif格式

实验图片路径为： imgs/exp1_1.jpg 输出路径为： outputs/ 请按照exp1_1_i的格式，输出四个结果，比如a)的结果保存为： outputs/exp1_1_1.jpg



```
1 # 函数定义
2 def add_numbers(a, b):
3     return a + b
4
5 # 调用函数
6 result = add_numbers(5, 7)
7 print("The sum is:", result)
```

图 6: 创建add_numbers.py脚本学习定义和调用函数



```
ouc@islouc-vm:~/Desktop$ nano add_numbers.py
ouc@islouc-vm:~/Desktop$ python3 add_numbers.py
The sum is: 12
```

图 7: 执行add_structures.py脚本

```
2. Matplotlib from PIL import Image
from pylab import *
import numpy as np
# 读取图像到数组中
im = np.array(Image.open('empire.jpg'))
# 绘制图像
imshow(im)
# 一些点
x = [100,100,400,400]
y = [200,500,200,500]
# 使用红色星状标记绘制点
plot(x,y,'r*')
# 绘制连接前两个点的线
plot(x[:2],y[:2])
# 添加标题，显示绘制的图像
title('Plotting: "empire.jpg"')
show()

3. Numpy
```

```

ouc@islouc-vm:~/Desktop$ nano diamond.py
ouc@islouc-vm:~/Desktop$ python3 diamond.py
*
***
*****
*****
*****
*****
*****
*

```

图 8: 输出菱形

```

GNU nano 4.8      loop.py
loop.py
def for_loop_example():
    """演示 for 循环"""
    print("For loop example:")
    for i in range(1, 6):
        print(f"Number: {i}")

def while_loop_example():
    """演示 while 循环"""
    print("\nWhile loop example:")
    i = 1
    while i <= 5:
        print(f"Number squared: {i**2}")
        i += 1

if __name__ == "__main__":
    for_loop_example()
    while_loop_example()

```

图 9: loop.py

4.1.imgs目录下有图像boardWithNoise.jpg，用Python写程序，采用自适应中值滤波器去除噪声干扰。

实验图片路径为： imgs/boardWithNoise.jpg

输出路径为： outputs/

请按照exp4_2_i的格式，输出每个任务结果

5.imgs目录下有图像windmill_noise.png，用Python写程序，去除条纹干扰。

实验图片路径为： imgs/windmill_noise.png

输出路径为： outputs/

请按照exp4_1_i的格式，输出每个任务结果

6.将Sobel算子编码到pytorch卷积核中，并用编码的卷积核对图像100_3228.jpg执行卷积操作，输出结果（水平梯度图像、垂直梯度图像和梯度幅值图像），理解卷积操作与空间域滤波的关系。

实验图片路径为： imgs/100_3228.jpg

输出路径为： outputs/

请按照exp2_3_i的格式，输出结果比如结果保存为： outputs/exp2_3_1.jpg

```

*
ouc@islouc-vm:~/Desktop$ nano loop.py
ouc@islouc-vm:~/Desktop$ python3 loop.py
For loop example:
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5

While loop example:
Number squared: 1
Number squared: 4
Number squared: 9
Number squared: 16
Number squared: 25

```

图 10: 执行loop.py

```

GNU nano 4.8 leap_year.py
def is_leap_year(year):
    """判断是否是闰年"""
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False

def main():
    """主函数"""
    year = int(input("请输入年份: "))
    if is_leap_year(year):
        print(f"{year} 是闰年。")
    else:
        print(f"{year} 不是闰年。")

if __name__ == "__main__":
    main()

```

图 11: leap_year.py

7.imgs目录下有图像laoshan.jpg，用Python写程序，将其作4阶haar小波变换，仅保留第四阶变换的系数，反变换，查看图像的结果。（Matlab代码已经给出，仅作参考）

实验图片路径为： imgs/laoshan.jpg

输出路径为： outputs/

请按照exp5_1_i的格式，输出每个任务结果

8.imgs目录下有图像1.jpg和2.jpg，用Python写程序，使用基于小波变换的方法将2.jpg中的人物融合到1.jpg中，提升融合效果。

实验图片路径为： imgs/1.jpg imgs/2.jpg

输出路径为： outputs/

请按照exp5_2_i的格式，输出每个任务结果

9.通过离散傅里叶变换我们可以得到频谱图，通过离散傅里叶逆变换我们可以将频谱图转换为原图，请使用pytorch实现离散傅里叶逆变换（可使用库函数或自定义函数），并将频谱图设置为初始值为高斯噪声的模型

```
ouc@islouc-vm:~/Desktop$ nano loop.py
ouc@islouc-vm:~/Desktop$ nano leap_year.py
ouc@islouc-vm:~/Desktop$ python3 leap_year.py
请输入年份: 2016
2016 是闰年。
```

图 12: 执行leap_year.py

```
1 from PIL import Image
2 image_1 = Image.open("imgs/exp1_1.jpg")
3 image_1.save("outputs/exp1_1_1.jpg")
4
5 image_2 = image_1.resize((640,480))
6 image_2.save("outputs/exp1_1_2.jpg")
7
8 image_3 = image_2.convert("L")
9 image_3.save("outputs/exp1_1_3.jpg")
10
11 image_3.save("outputs/exp1_1_4.gif")
```

图 13: PIL库运用实例

参数，利用逆变换的结果与原图之间的均方误差作为损失函数对模型参数进行优化，验证是否能够通过优化学习到频谱图。

实验图片路径为: imgs/2.JPG

输出路径为: outputs/

请按照exp3_3_i的格式，输出结果

10.墙纸分割实验

1.3 命令行环境

1.我们可以使用类似 `ps aux | grep` 这样的命令来获取任务的 pid，然后您可以基于pid 来结束这些进程。但我们其实有更好的方法来做这件事。在终端中执行 `sleep 10000` 这个任务。然后用 `Ctrl-Z` 将其切换到后台并使用 `bg`来继续允许它。现在，使用 `pgrep` 来查找 pid 并使用 `pkill` 结束进程而不需要手动输入pid。(提示：：使用 `-af` 标记)。

`sleep 10000`

`Ctrl-Z`

`bg`

`pgrep sleep #列出包含关键字 sleep 的进程的 pid`



图 14: a)输出结果



图 15: b)输出结果

```
# 76560
```

```
# 81186
```

```
# 81292
```

```
pgrep sleep 10000 #列出包含关键字 sleep 的进程的 pid
```

```
>
```

```
# 76560
```

```
# 81186
```

```
# 81292
```

-a Include process ancestors in the match list. By default, the current pgrep or pkill process and all of its ancestors are excluded (unless -v is used).

-f Match against full argument lists. The default is to match against process names.

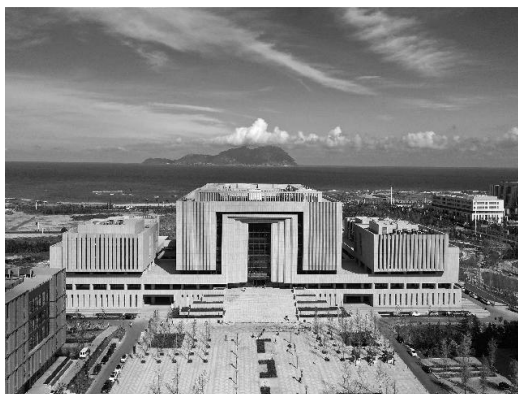


图 16: c)输出结果

```
huc@islouc-vm:~/Desktop$ nano matplotlib.py
huc@islouc-vm:~/Desktop$ vim matplotlib.py
Error detected while processing /home/ouc/.vimrc:
line 15:
E492: Not an editor command: 语法高亮
line 48:
E492: Not an editor command: 然后把它保存到 ~/.vimrc! Comments in Vimscript start with a `
`.
Press ENTER or type command to continue
huc@islouc-vm:~/Desktop$ python3 matplotlib.py
Traceback (most recent call last):
  File "matplotlib.py", line 2, in <module>
    pil_im = Image.open('empire.jpg')
  File "/usr/lib/python3/dist-packages/PIL/Image.py", line 2809, in open
    fp = builtins.open(filename, "rb")
FileNotFoundError: [Errno 2] No such file or directory: 'empire.jpg'
```

图 17: 运行matplotlib.py

`kill -af sleep`

2.如果您希望某个进程结束后再开始另外一个进程，应该如何实现呢？在这个练习中，我们使用 `sleep 60 &` 作为先执行的程序。一种方法是使用 `wait` 命令。尝试启动这个休眠命令，然后待其结束后再执行 `ls` 命令。 `sleep 60 & pgrep sleep | wait; ls` 但是，如果我们在不同的 `bash` 会话中进行操作，则上述方法就不起作用了。因为 `wait` 只能对子进程起作用。之前我们没有提过的一个特性是，`kill` 命令成功退出时其状态码为 `0`，其他状态则是非`0`。`kill -0` 则不会发送信号，但是会在进程不存在时返回一个不为`0`的状态码。请编写一个 `bash` 函数 `pidwait`，它接受一个 `pid` 作为输入参数，然后一直等待直到该进程结束。您需要使用 `sleep` 来避免浪费 CPU 性能。

```
pidwait()
{ while kill -0 $1 #循环直到进程结束
do
sleep 1
done
```

```
GNU nano 4.8 numpy.py
from PIL import Image
from numpy import *
import numpy as np
im = np.array(Image.open('empire.jpg').convert('L'))
im2 = 255 - im # 对图像进行反相处理
im3 = (100.0/255) * im + 100 # 将图像像素值变换到 100...200 区间
im4 = 255.0 * (im/255.0)**2 # 对图像像素值求平方后得到的图像
```

图 18: numpy.py

```
python3 numpy.py
Traceback (most recent call last):
  File "numpy.py", line 2, in <module>
    from numpy import *
  File "/home/ouc/Desktop/numpy.py", line 4, in <module>
    im = np.array(Image.open('empire.jpg').convert('L'))
AttributeError: partially initialized module 'numpy' has no attribute 'array' (most likely
due to a circular import)
```

图 19: 运行numpy.py

ls }

需要注意的是，这里 while 判断的是命令行的返回值而不是布尔值，这个和其他语言有所区别。返回值 0 表示成功所以能够进入循环

```
sleep 60 & pidwait $(pgrep sleep 60) [1] 554
```

```
[1] + 554 done sleep 60
```

```
pidwait:kill:2: kill 554 failed: no such process
```

```
buggy.sh debug_for.sh html_root out.log
```

```
debug.sh html.zip marco.sh
```

3.创建一个 dc 别名，它的功能是当我们错误的将 cd 输入为 dc 时也能正确执行。

```
alias dc=cd
```

4.执行 history | awk '{ \$1=""; print substr(\$0,2) }' | sort | uniq -c | sort -n | tail -n 10 来获取您最常用的十条命令，尝试为它们创建别名。注意：这个命令只在 Bash 中生效，如果您使用 ZSH，使用history 1 替换 history。

```
history | awk ' $1=""; print substr($0,2) ' | sort | uniq -c | sort -n | tail -n 10
```

```
62 g++ -std=c++11 tests.cpp
```

```
64 npm run src:build
```

```
68 git pull
```

```
80 ./a.out
```

```
118 git status
```

```
186 ll
```

```

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

# 显示中文及设置坐标轴字体
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

def set_font_size(size=8):
    """设置坐标轴字体大小"""
    plt.xticks(fontsize=size)
    plt.yticks(fontsize=size)

def adaptive_median_filter(img, max_size=7, border_type=cv.BORDER_REFLECT):
    """自适应中值滤波算法"""
    rows, cols = img.shape
    denoised = np.zeros_like(img)
    padded = cv.copyMakeBorder(img, max_size, max_size, max_size, max_size, border_type)

    for i in range(rows):
        for j in range(cols):
            window = padded[i:i+2*max_size+1, j:j+2*max_size+1]
            denoised[i, j] = apply_filter(window, max_size, img[i, j])

    return denoised

def apply_filter(window, max_size, pixel):
    """递归中值滤波"""
    size = 3
    median = np.median(window)
    min_val, max_val = np.min(window), np.max(window)

    if min_val < median < max_val:
        if min_val < pixel < max_val:
            return pixel
        else:
            return median
    elif size < max_size:
        padded = cv.copyMakeBorder(window, 1, 1, 1, 1, cv.BORDER_REFLECT)
        return apply_filter(padded, max_size, pixel)
    else:
        return median

if __name__ == '__main__':
    img_noisy = cv.imread('imgs/boat-dilt-noise.jpg', 0) # 读取含噪灰度图
    img_denoised = adaptive_median_filter(img_noisy)
    img_median = cv.medianBlur(img_noisy, 7)

    plt.figure(figsize=(12, 4))
    plt.subplot(131), plt.imshow(img_noisy, 'gray'), plt.title('原始噪声图像'), set_font_size(12)
    plt.subplot(132), plt.imshow(img_denoised, 'gray'), plt.title('自适应中值滤波'), set_font_size(12)
    plt.subplot(133), plt.imshow(img_median, 'gray'), plt.title('标准中值滤波'), set_font_size(12)
    plt.tight_layout()
    plt.savefig('outputs/exp4_2.1.jpg', dpi=300, bbox_inches='tight')
    plt.show()

```

图 20: 自适应中值滤波去噪上半部分

```

size = 3
median = np.median(window)
min_val, max_val = np.min(window), np.max(window)

if min_val < median < max_val:
    if min_val < pixel < max_val:
        return pixel
    else:
        return median
elif size < max_size:
    padded = cv.copyMakeBorder(window, 1, 1, 1, 1, cv.BORDER_REFLECT)
    return apply_filter(padded, max_size, pixel)
else:
    return median

if __name__ == '__main__':
    img_noisy = cv.imread('imgs/boat-dilt-noise.jpg', 0) # 读取含噪灰度图
    img_denoised = adaptive_median_filter(img_noisy)
    img_median = cv.medianBlur(img_noisy, 7)

    plt.figure(figsize=(12, 4))
    plt.subplot(131), plt.imshow(img_noisy, 'gray'), plt.title('原始噪声图像'), set_font_size(12)
    plt.subplot(132), plt.imshow(img_denoised, 'gray'), plt.title('自适应中值滤波'), set_font_size(12)
    plt.subplot(133), plt.imshow(img_median, 'gray'), plt.title('标准中值滤波'), set_font_size(12)
    plt.tight_layout()
    plt.savefig('outputs/exp4_2.1.jpg', dpi=300, bbox_inches='tight')
    plt.show()

```

图 21: 自适应中值滤波去噪下半部分

258 cd ..

263 git push

376 git add .

1066 ls

2 解题感悟

通过这次试验我了解了python基础知识和python计算机视觉相关知识，了解了如何在虚拟机中创建以及运行python脚本，了解了numpy、PIL、matplotlib、opencv等python库。此外还发现如果Tex course中图片太多时可以保存到Tex course里，无法粘贴也没事，同时也发现他们环境确实不太稳定，执行了命令之后环境可能会改变导致之前可以执行成功的命令无法执行。

```

import cv2
import numpy as np
# 读取图像
image_path = "imgs/windmill_noise.png"
output_path = "outputs/"
# 读取图像
img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
# 傅里叶变换
f = np.fft.fft2(img)
fshift = np.fft.fftshift(f)
# 频谱图
magnitude_spectrum = 20 * np.log(np.abs(fshift))
# 去除条纹干扰
rows, cols = img.shape
crow, ccol = rows // 2, cols // 2
# 创建矩形窗口
mask = np.ones((rows, cols), np.uint8)
mask[crow-30:crow+30, ccol-30:ccol+30] = 0
# 应用掩码和逆傅里叶变换
fshift = fshift * mask
f_ishift = np.fft.ifftshift(fshift)
img_back = np.fft.ifft2(f_ishift)
img_back = np.abs(img_back)
# 反转颜色以增加对比度
img_back = 255 - img_back
# 保存结果
output_file = output_path + "exp4_1_1_windmill_noise_removed.png"
cv2.imwrite(output_file, img_back)

```

图 22: 去除条纹干扰

```

import torch.nn as nn
from PIL import Image
import numpy as np
import torchvision.transforms as transforms
from torch.autograd import Variable

# 加载图像
image_path = "imgs/100_3228.jpg"
image = Image.open(image_path).convert('L') # 转换为灰度图像
transform = transforms.ToTensor()
image_tensor = transform(image)
image_tensor = image_tensor.unsqueeze(0) # 添加batch维度

# 定义Sobel算子的水平和垂直卷积核
sobel_horizontal = np.array([[[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]], dtype=np.float32)
sobel_vertical = np.array([[[-1, -2, -1], [0, 0, 0], [1, 2, 1]], dtype=np.float32)

# 创建PyTorch卷积层
sobel_horizontal_conv = nn.Conv2d(1, 1, kernel_size=3, bias=False, padding=1)
sobel_vertical_conv = nn.Conv2d(1, 1, kernel_size=3, bias=False, padding=1)

# 设置卷积核权重
sobel_horizontal_conv.weight.data = torch.from_numpy(sobel_horizontal).view(1, 1, 3, 3)
sobel_vertical_conv.weight.data = torch.from_numpy(sobel_vertical).view(1, 1, 3, 3)

# 对图像应用水平和垂直Sobel滤波器
horizontal_gradient = sobel_horizontal_conv(Variable(image_tensor))
vertical_gradient = sobel_vertical_conv(Variable(image_tensor))

```

图 23: 卷积

3 GitHub链接

<https://github.com/zyx-cyber/coursecontent.git>

```

import pywt
import numpy as np
from PIL import Image

# 读取图像并转换为灰度图
image_path = 'imgs/laoshan.jpg'
image = Image.open(image_path).convert('L')
image_array = np.array(image)

# 执行4级Haar小波变换
coeffs = pywt.wavedec2(image_array, 'haar', level=4)

# 仅使用最高层次的系数进行逆变换
coeffs[1:] = [tuple([np.zeros_like(v) for v in subband]) for subband in coeffs[1:]]
reconstructed_image_array = pywt.waverec2(coeffs, 'haar')

# 确保数据范围在0-255之间，并转换为uint8类型
reconstructed_image_array = np.clip(reconstructed_image_array, 0, 255).astype(np.uint8)

# 转换为图像并保存
output_path = 'outputs/exp5_1_1.jpg'
reconstructed_image = Image.fromarray(reconstructed_image_array)
reconstructed_image.save(output_path)

```

图 24: 小波变换

```

import numpy as np
import cv2 as cv

def fangcha(img):
    row, col = img.shape
    varImg = np.zeros((row, col))

    padded_img = np.pad(img, ((5, 5), (5, 5)), mode='reflect')

    for i in range(row):
        for j in range(col):
            window = padded_img[i:i+11, j:j+11]
            var = cv.meanStdDev(window)
            varImg[i, j] = var

    return varImg

def qiuquan(img1, img2):
    array1 = fangcha(img1)
    array2 = fangcha(img2)

    denom = array1 + array2
    weight1 = np.divide(array1, denom, out=np.zeros_like(array1), where=denom!=0)
    weight2 = np.divide(array2, denom, out=np.zeros_like(array2), where=denom!=0)

    return weight1, weight2

```

图 25: 小波变换上半部分

```

    return weight1, weight2

def ronghe(img1, img2):
    b, g, r = cv.split(img1)
    b1, g1, r1 = cv.split(img2)

    weight_b1, weight_b2 = qiuquan(b, b1)
    weight_g1, weight_g2 = qiuquan(g, g1)
    weight_r1, weight_r2 = qiuquan(r, r1)

    b_fused = (weight_b1 * b + weight_b2 * b1).astype(np.uint8)
    g_fused = (weight_g1 * g + weight_g2 * g1).astype(np.uint8)
    r_fused = (weight_r1 * r + weight_r2 * r1).astype(np.uint8)

    new_img = cv.merge([b_fused, g_fused, r_fused])
    return new_img

img1 = cv.imread("imgs/1.jpg")
img2 = cv.imread("imgs/2.jpg")
img3 = ronghe(img1, img2)
cv.imwrite("outputs/exp5_2_1.jpg", img3)

```

图 26: 小波变换下半部分

```

import torch
import torch.nn.functional as F
from PIL import Image
from torchvision import transforms

# 读取图片
image_path = "imgs/2.JPG"
output_path = "outputs/"
image = Image.open(image_path).convert('L') # 转换为灰度图
image.save(output_path + 'exp3_3_i_original.jpg')

# 转换为张量
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5], std=[0.5]) # 归一化到[-1, 1]
])
image_tensor = transform(image).unsqueeze(0) # 添加Batch维度

# 计算DFT
spectrum = torch.fft.fftshift(torch.fft.fft2(image_tensor))

# 初始化带有高斯噪声的频谱图
noise_std = 0.1 # 噪声标准差, 可根据需要调整
noisy_spectrum = spectrum + torch.randn_like(spectrum) * noise_std
torch.save(noisy_spectrum, output_path + 'exp3_3_i_noisy_spectrum.pt')

# 定义逆变换函数
def inverse_fft(noisy_spectrum):

```

图 27: 离散傅里叶逆变换上半部分

```

# 定义均方误差损失函数
def mse_loss(image1, image2):
    return F.mse_loss(image1, image2)

# 初始化优化器
optimizer = torch.optim.Adam([noisy_spectrum], lr=0.01)

# 迭代优化
num_iterations = 1000 # 迭代次数, 可根据需要调整
for i in range(num_iterations):
    # 使用逆变换函数对噪声频谱进行逆变换
    reconstructed_image = inverse_fft(noisy_spectrum)

    # 计算损失
    loss = mse_loss(reconstructed_image, image_tensor)

    # 反向传播和优化
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    if (i + 1) % 100 == 0:
        print(f"Iteration {i + 1}/{num_iterations}, Loss: {loss.item()}")

# 保存结果
reconstructed_image = inverse_fft(noisy_spectrum)
image.fromarray((reconstructed_image.squeeze(0).numpy() * 255).astype('uint8')).save(output_path + 'exp3_3_i_reconstructed_image.jpg')

```

图 28: 离散傅里叶逆变换下半部分

```

import cv2
import numpy as np
from matplotlib import pyplot as plt
from skimage import exposure

# 定义函数进行图像处理
def process_image(image_path):
    # 读取并显示原图
    image = cv2.imread(image_path)
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    plt.figure(figsize=(10, 10))
    plt.imshow(image_rgb)
    plt.title('Original Image')
    plt.axis('off')
    plt.show()

    # 转换为灰度图像
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    # 使用高斯滤波器对灰度图像进行平滑
    gray = exposure.equalize_adapthist(gray, clip_limit=0.03)
    # 使用 Otsu 方法进行二值化
    ret, binary = cv2.threshold(gray * 255, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

    # 显示二值化结果
    plt.figure(figsize=(10, 10))
    plt.imshow(binary, cmap='gray')
    plt.title('Binary Image')
    plt.axis('off')
    plt.show()

```

图 29: 墙纸分割实验上半部分

```

plt.title('Binary Image')
plt.axis('off')
plt.show()
# 找到轮廓
contours, hierarchy = cv2.findContours(binary, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
# 创建一个空白图像用于绘制轮廓
foreground = np.zeros_like(image_rgb)
# 绘制前景
cv2.drawContours(foreground, contours, -1, (255, 255, 255), thickness=cv2.FILLED)
# 显示前景图像
plt.figure(figsize=(10, 10))
plt.imshow(foreground)
plt.title('Foreground Image')
plt.axis('off')
plt.show()
return foreground

```

图像路径列表

```

image_paths = ['D:/wallpaper/wallpaper1.jpg.jpg', 'D:/wallpaper/wallpaper2.jpg.jpg', 'D:/wallpaper/wallpaper3.jpg.jpg', 'D:/wallpaper/wallpaper4.jpg.jpg', 'D:/wallpaper/wallpaper5.jpg.jpg']

```

循环处理并显示图像

```

for image_path in image_paths:
    processed_image = process_image(image_path)

```

图 30: 墙纸分割实验下半部分

```

ouc@islouc-vm:~/Desktop$ sleep 10000
^Z
[3]+  Stopped                  sleep 10000
ouc@islouc-vm:~/Desktop$ bg
[3]+  sleep 10000 &
ouc@islouc-vm:~/Desktop$ pgrep sleep
9678
9716
10300
ouc@islouc-vm:~/Desktop$ pgrep sleep 10000
pgrep: only one pattern can be provided
Try 'pgrep --help' for more information.
ouc@islouc-vm:~/Desktop$ -a
-a: command not found
ouc@islouc-vm:~/Desktop$ -f
-f: command not found
ouc@islouc-vm:~/Desktop$ pkill -af sleep
pkill: invalid option -- 'a'

```

图 31: 获取任务pid及相关操作

```

ouc@islouc-vm:~/Desktop$ sleep 60 &
[4] 10328
ouc@islouc-vm:~/Desktop$ pgrep sleep | wait;
ouc@islouc-vm:~/Desktop$ ls
add_numbers.py      debug_for.sh        marco.sh            result.txt
add_structures.py   diamond.py           matplotlib.py       tall-trees-wyql.sh
A.txt               chakra.desktop      misc               test
bigexp.py           html_root            numpy.py            test.c
B.txt               html.zip             numpy.py            tap
class               IDA Framework 8.3.desktop  out.log             variables.py
control_structures.py leap_year.py         pmaptest.sh
data                loop.py              __pycache__
ouc@islouc-vm:~/Desktop$ vim pidwait.sh
Error detected while processing /home/ouc/.vimrc:
line 15:
E492: Not an editor command: 语法高亮
line 48:
E492: Not an editor command: 然后把它保存到 ~/.vimrc! Comments in Vimscript start with a ""
Press ENTER or type command to continue
[4]+  Done                  sleep 60
ouc@islouc-vm:~/Desktop$ sleep 60 & pidwait $(pgrep sleep 60)
[4] 10388
pgrep: only one pattern can be provided
Try 'pgrep --help' for more information.
pidwait: command not found
ouc@islouc-vm:~/Desktop$ source pidwait.sh
ouc@islouc-vm:~/Desktop$ sleep 60 & pidwait $(pgrep sleep 60)
[5] 10451
pgrep: only one pattern can be provided
Try 'pgrep --help' for more information.
kill: usage: Kill [-s sigspec | -n signum | -s sigspec] pid | jobspec ... or kill -l [sigspec]

```

图 32: 某进程结束后开始另一进程

```

ouc@islouc-vm:~/Desktop$ alias dc=cd
[5]+  Done                  sleep 60
ouc@islouc-vm:~/Desktop$ dc

```

图 33: 创建别名

```

marco@kali:~$ history | awk '{if($1!="");print substr($0,2)}' | sort | uniq -c | sort -n | tail
n 10
 5 ls
 5 make
 5 vim matplotlib.py
 6 checksec --file test
 6 stull6@10.140.32.159 -p 47116
 6 ./test
 7 python3 matplotlib.py
11 #/bin/bash
59 ssh-keygen -R "[10.140.32.159]:47116"
61 ssh stull6@10.140.32.159 -p 47116

marco@kali:~$ sudo ls
BurpSuiteCommunity Downloads last.modified.txt Music Public Videos
Desktop ghidra_10.3.3_PUBLIC last.modified.txt Pictures snap
Documents idafree-8.3 marco_history.log pidwait.sh Templates

```

图 34: 获取最常用的10条命令，尝试为他们创建别名