计算机视觉与模式识别
实验一

指导老师:李慧

专业:**22 级计算机科学与技术 1 班**
学号:**20224001027**
学生:**何泽铭**

2025 年 3 月 10 日

目录

1 实验目的

1.1 统计化学元素数目

统计每幅图像中化学元素（Al、Fe、P）的分布数目。通过有效的分析方法，确定各元素在图像中的数量，为后续研究提供基础数据。

1.2 两两元素重叠分析

1. 统计 Al 和 Fe 元素的重叠数目，明确这两种元素在图像中共同出现的区域和数量。
2. 统计 Fe 和 P 元素的重叠数目，了解这两种元素的重叠情况，揭示元素间可能的关联。
3. 统计 Al 和 P 元素的重叠数目，分析它们在图像中的重叠特征。
4. 以图像形式展示两两元素的重叠情况，直观呈现元素重叠的位置和范围，便于进一步观察和分析。

1.3 三者元素重叠分析

1. 统计 Al、Fe 和 P 元素三者重叠的数目，确定三种元素在图像中同时出现的区域的数量。
2. 以图像形式展示三种元素的重叠情况，通过可视化的方式呈现复杂的元素重叠信息，帮助理解元素间的相互作用。

1.4 实验拓展

本实验方法应具备颜色无关性，即无论原始图像中化学元素的颜色如何，均可实现上述实验目标。这一特性确保了实验方法的通用性和可靠性，能够适应不同来源和特点的图像数据。

2 实验内容

2.1 实验概述

本实验旨在分析三幅图像（AI.jpg、Fe.jpg、P.jpg），这些图像是通过成像技术得到的不同化学元素的分布情况。实验目标是统计每种元素的数量，计算两种及三种元素重叠的数量，并将这些结果以图像的形式展示出来。

2.2 材料与方法

2.2.1 材料

包含三个文件的 `image` 文件夹：

- AI.jpg
- Fe.jpg
- P.jpg

2.2.2 方法

使用 OpenCV 加载图像，通过将 RGB 图像转换为 HSV 色彩空间来处理，然后应用阈值处理将图像二值化，以便于后续的元素计数和重叠分析。使用位运算（AND 操作）确定不同元素之间的重叠区域。统计每个单独图像中的非零像素点数以获取元素数量。生成新的图像显示两两元素及三者重叠的情况。

3 实验过程

计算机视觉常用的方法是使用 OpenCV 的库函数对图像进行处理。因此我们先利用 `cv2.imread()` 函数读入图像数据。本例子中，共拥有 3 张图片，分别是 Al，Fe 和 P（即铝、铁、磷）三种化学元素。

读取图片的代码如下：

```
# 设置图片路径
Al_path = os.path.join("image", "AI.jpg")
Fe_path = os.path.join("image", "Fe.jpg")
P_path = os.path.join("image", "P.jpg")
# print("图片形状:", al.shape)

# 读取三张图片
Al = cv2.imread(Al_path)
Fe = cv2.imread(Fe_path)
P = cv2.imread(P_path)
```

注意，这里的路径最好不要直接写死，而是用 `python os` 库中提供的 `os.path.join()` 函数进行拼接处理。该函数能够自行判断当前运行环境而选择路径的分隔符，无论是 Linux 还是 Windows 都不会出错。

读入图片后，我发现图片中的左上角 logo 处和左下角的 50nm 标识都不是我需要的东西，需要剔除。因此需要对输入图像进行预处理：

- 为了去除左上角的 logo，根据观察可以看到，logo 读取进来的图像矩阵所在的行数均在 74 行及以内，因此我们处理的时候可以针对 logo 所在的行数进行删除。
- 为了去除左下角的 50nm 标识，我们需要识别出白色区域并去除。

为了实现上面的目标，我先将图片从 RGB 色彩通道转换为 HSV(HSB) 色彩通道。OpenCV 的库函数也提供了相应的转换。我们平常所见的图片绝大多数都是 RGB 通道，即利用 Red（红）、Green（绿）、Blue（蓝）三种颜色及其浓度来表示一张彩色图片。在 8Bit 色深的 RGB 图像中，RGB 三个通道的取值区间均为 [0, 255]。而 HSV 色彩通道是利用 Hue（色调）、Saturation（饱和度）、Value（明度）三值来表示颜色。不同于 RGB 图像，HSV 图像是根据色调 H 来确定颜色的，而 RGB 图像一般需要通过三个值共同参与来表达一种颜色。

处理代码如下：

```
# RGB2HSV
def rgb2hsv(img):
    hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    h, s, v = cv2.split(hsv_img)
    return [h, s, v]
```

显示 HSV 图像三通道的代码如下：

```
# Plot HSV image
def plot_hsv_image(img: list):
    cv2.imshow('H', img[0])
    cv2.imshow('S', img[1])
```

```
    cv2.imshow('V', img[2])
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

未经处理前，Al 的分拆 HSV 通道后的图像如下：

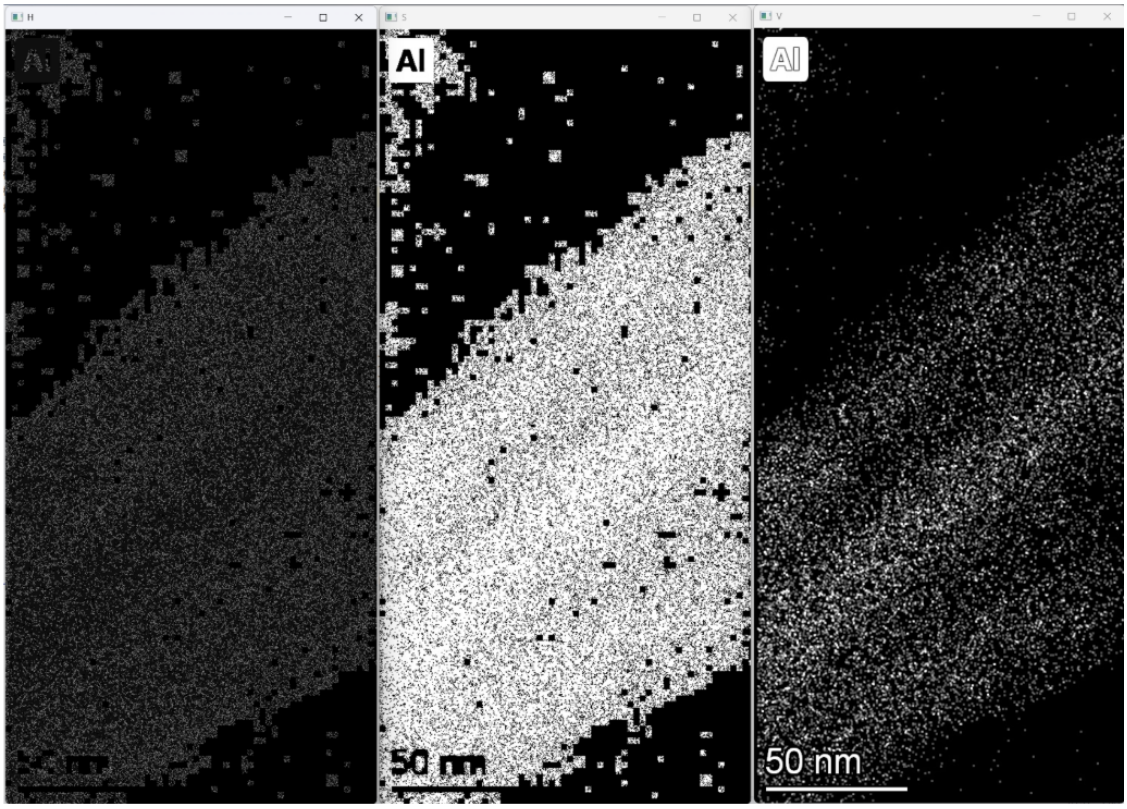

图 1: Al 的分拆 HSV 通道后的图像

在 HSV 图像中，查询可知黑色点范围如下：

- H: [0, 180]
- S: [0, 255]
- V: [0, 46]

白色点范围如下：

- H: [0, 180]
- S: [0, 30]
- V: [221, 255]

logo 区域的像素点较为明亮，因此可以直接利用位置和 V 的值进行判断去除。

图像预处理代码如下：

```python
# 完成实验拓展要求，去除 logo 区域 和 黑白点
def remove_LogoAndPoints(split_img: list):
    h, s, v = split_img
    for i in range(h.shape[0]):
        for j in range(h.shape[1]):
            # 去除黑色像素点
            if 0 <= h[i, j] <= 180 and 0 <= v[i, j] <= 46:
                h[i, j] = 0
                s[i, j] = 0
                v[i, j] = 0
            # 去除白色像素点
            elif 0 <= h[i, j] <= 180 and 221 <= v[i, j] <= 255 and 0 <= s[i, j] <= 30:
                h[i, j] = 0
                s[i, j] = 0
                v[i, j] = 0
            # 去除 logo 区域
            if i <= 70 and 180 <= v[i, j] <= 255:
                h[i, j] = 0
                s[i, j] = 0
                v[i, j] = 0
    return [h, s, v]
```

统计元素量时，为了方便统计，我把有颜色的像素点都统计为 1 个元素。而我已在上一步去除了杂色点，因此这一步可以直接数图像 V 通道的点数进行统计。**统计函数如下：**

```python
# 统计各化学元素的数目
def count_points(img: list):
    h, s, v = img
    cnt = 0
    for i in range(v.shape[0]):
        for j in range(v.shape[1]):
            if v[i, j] != 0:
                cnt += 1
    return cnt
```

统计重叠量时，依旧是利用上一步的思想进行统计，但是这时候我需要两张图片的交集，即需要多加一个“与”的判断步骤。为了显示图片，还需要返回合并后的 HSV 矩阵。**处理函数如下：**

```python
# 求解重叠量，并将元素两两的重叠量画出来
def cnt2overlap(img1: list, img2: list):
    h1, s1, v1 = img1
    h2, s2, v2 = img2
    # 找到两者重叠的区域
    overlap_mask = (v1 != 0) & (v2 != 0)
    # 计算重叠量
    cnt = np.sum(overlap_mask)
```

```
    # 计算 h0, s0, v0
    h0 = np.zeros_like(h1, dtype=np.uint8)
    s0 = np.zeros_like(s1, dtype=np.uint8)
    v0 = np.zeros_like(v1, dtype=np.uint8)
    h0[overlap_mask] = np.clip(h1[overlap_mask] + h2[overlap_mask], 0, 255)
    s0[overlap_mask] = np.clip(s1[overlap_mask] + s2[overlap_mask], 0, 255)
    v0[overlap_mask] = np.clip(v1[overlap_mask] + v2[overlap_mask], 0, 255)
    return cnt, [h0, s0, v0]
```

三张图片的元素重叠量也是同理。**处理函数如下：**

```
# 同理，求三者的重叠量
def cnt3overlap(img1: list, img2: list, img3: list):
    h1, s1, v1 = img1
    h2, s2, v2 = img2
    h3, s3, v3 = img3
    # 找到三者重叠的区域
    overlap_mask = (v1 != 0) & (v2 != 0) & (v3 != 0)
    # 计算重叠量
    cnt = np.sum(overlap_mask)
    # 计算 h0, s0, v0
    h0 = np.zeros_like(h1, dtype=np.uint8)
    s0 = np.zeros_like(s1, dtype=np.uint8)
    v0 = np.zeros_like(v1, dtype=np.uint8)
    h0[overlap_mask] = np.clip(h1[overlap_mask] + h2[overlap_mask] + h3[overlap_mask
        ], 0, 255)
    s0[overlap_mask] = np.clip(s1[overlap_mask] + s2[overlap_mask] + s3[overlap_mask
        ], 0, 255)
    v0[overlap_mask] = np.clip(v1[overlap_mask] + v2[overlap_mask] + v3[overlap_mask
        ], 0, 255)
    return cnt, [h0, s0, v0]
```

至此，实验目标已经全部完成。由于我将重要的功能封装成函数且与具体颜色无关，因此无论原始图像中的化学元素是何种颜色，只需修改图像路径，均可实现本目标。

4 实验结果

```
(DL) PS D:\doc\study\大三下\计算机视觉与模式识别\EXP1> & D:/miniforge3/
铝元素的点数为： 109393
铁元素的点数为： 31416
磷元素的点数为： 12593
铝和铁的重叠量为： 10891
铝和磷的重叠量为： 5009
铁和磷的重叠量为： 1595
铝、铁、磷三者的重叠量为： 746
(DL) PS D:\doc\study\大三下\计算机视觉与模式识别\EXP1>
```

图 2: 实验结果

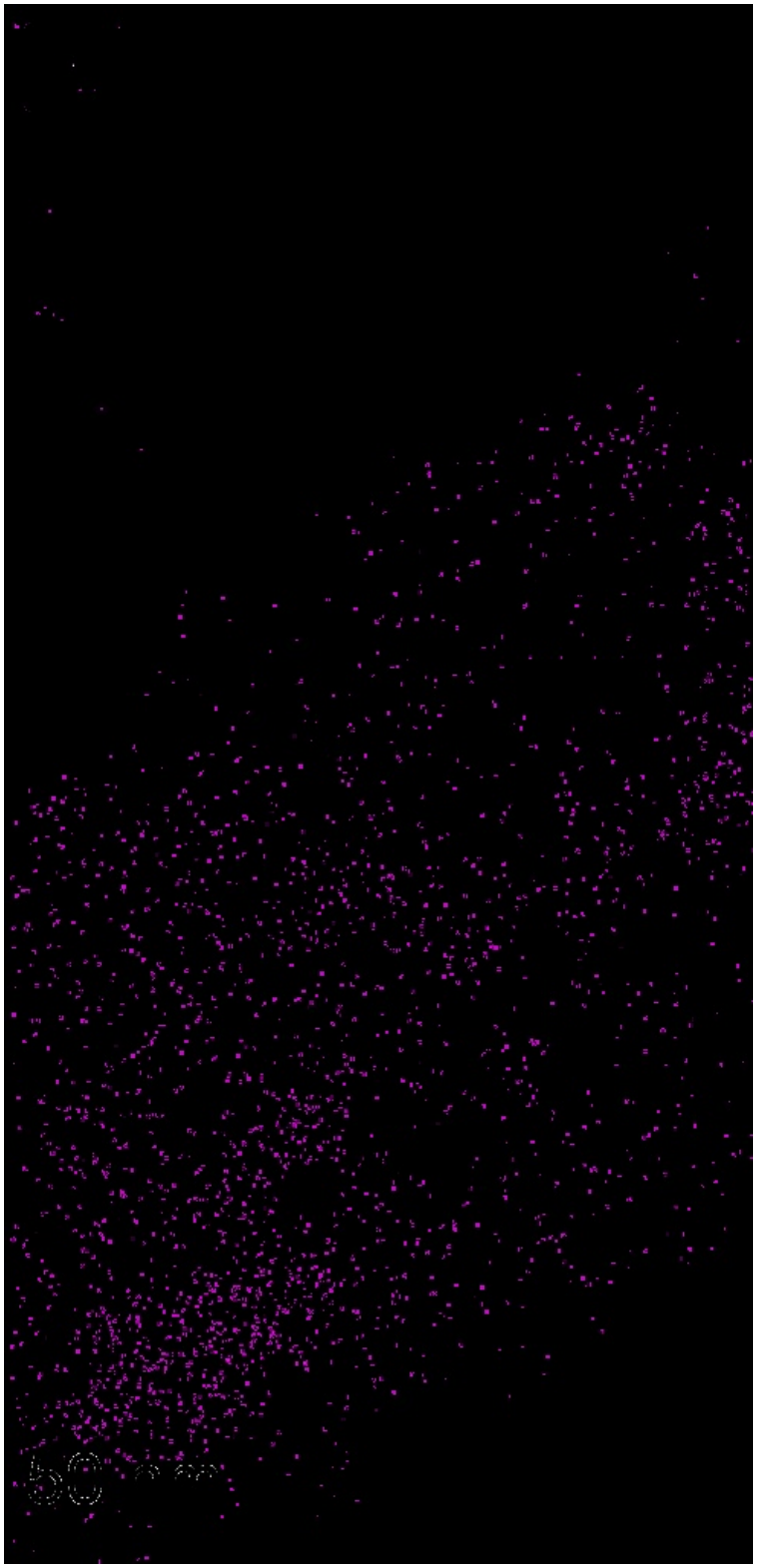

图 3: Al 和 Fe 的重叠图片

图 4: Al 和 P 的重叠图片

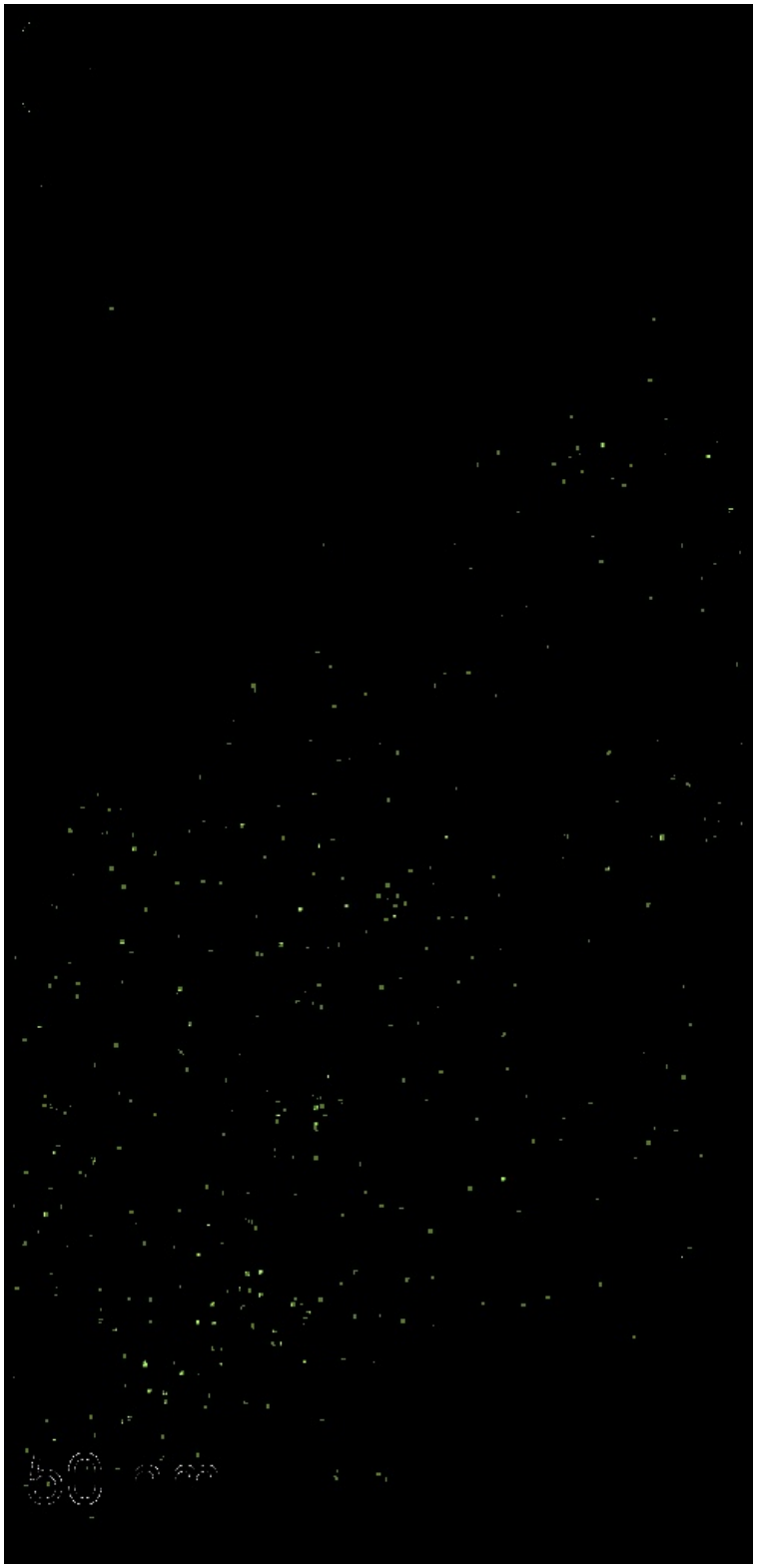

图 5: Fe 和 P 的重叠图片

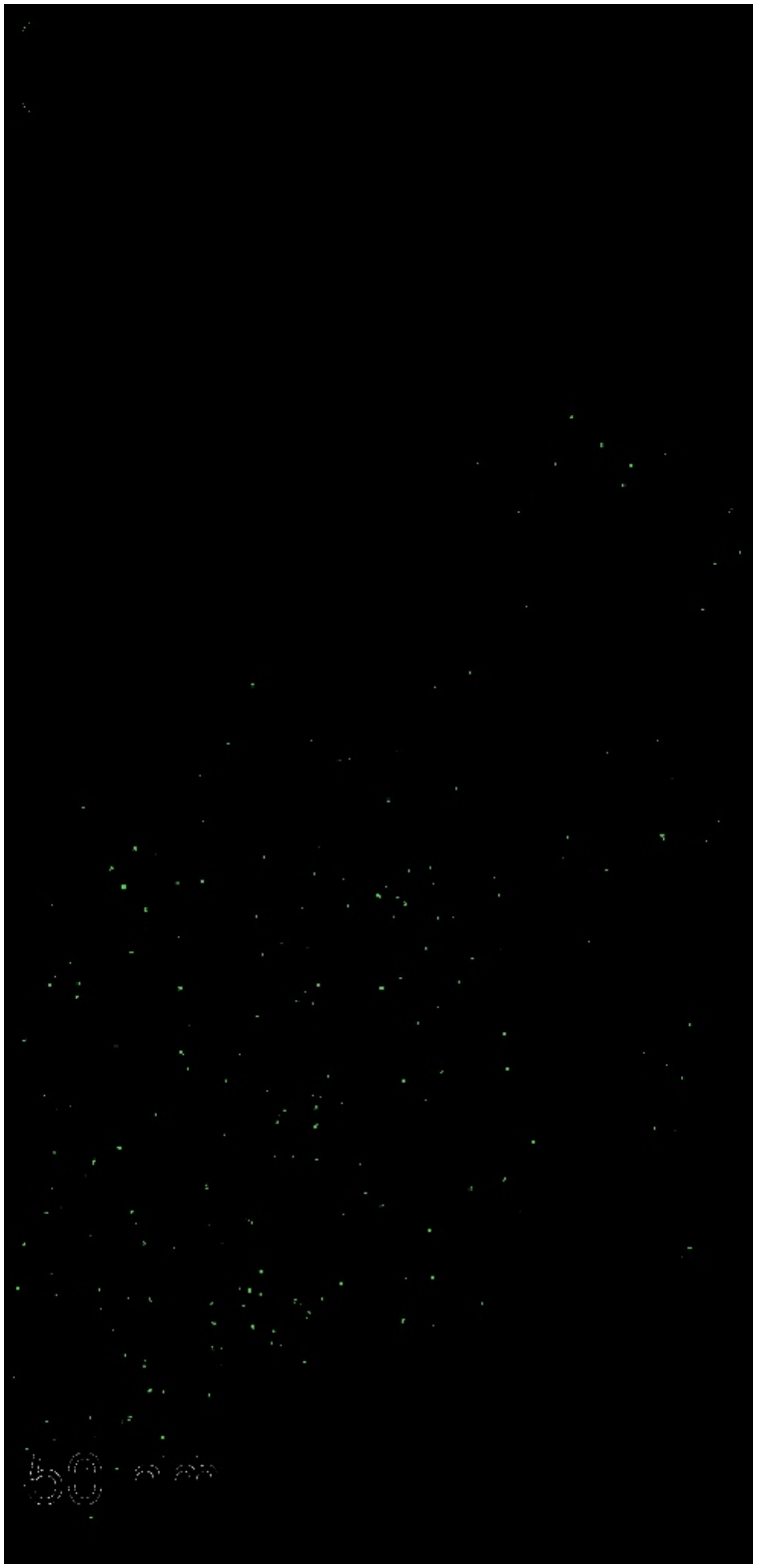

图 6: 三者重叠图片

5 实验小结

这次实验让我对图像处理和化学元素分布分析有了更深入的理解。通过实际操作，我学会了如何使用 OpenCV 库读取和处理图像，并将 RGB 图像转换为 HSV 空间，以便更好地分离颜色信息。实验中最有趣的部分是通过像素值的判断来去除干扰区域（如 logo 和黑白点），这让我意识到图像预处理在实际应用中的重要性。

在统计元素数目和重叠区域时，我遇到了一些挑战，比如如何高效地遍历像素并判断重叠条件。通过将功能封装成函数，我不仅提高了代码的可读性，还增强了代码的复用性。最终，我成功实现了两两元素和三元素的重叠分析，并将结果以图像形式展示出来，这让我感到非常有成就感。

这次实验让我深刻体会到，图像处理不仅仅是简单的像素操作，还需要结合具体的应用场景进行优化。虽然过程中遇到了一些问题，但通过查阅资料和调试代码，我逐步解决了这些问题，这让我对编程和图像处理有了更强的信心。希望以后能有更多类似的机会，继续探索计算机视觉的奥秘！

6 完整代码

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

# 设置图片路径
Al_path = os.path.join("image", "Al.jpg")
Fe_path = os.path.join("image", "Fe.jpg")
P_path = os.path.join("image", "P.jpg")
# print("图片形状:", al.shape)

# 读取三张图片
Al = cv2.imread(Al_path)
Fe = cv2.imread(Fe_path)
P = cv2.imread(P_path)

# RGB2HSV
def rgb2hsv(img):
    hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    h, s, v = cv2.split(hsv_img)
    return [h, s, v]

# Plot image
def plot_3image(img: list):
    plt.figure(figsize=(15, 15))
    for i, img in enumerate(img):
        plt.subplot(1, 3, i + 1)
        plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
        plt.axis("off")
        plt.show()

```

```python
# Plot HSV image
def plot_hsv_image(img: list):
    cv2.imshow('H', img[0])
    cv2.imshow('S', img[1])
    cv2.imshow('V', img[2])
    cv2.waitKey(0)
    cv2.destroyAllWindows()

# 完成实验拓展要求，去除 logo 区域 和 黑白点
def remove_LogoAndPoints(split_img: list):
    h, s, v = split_img
    for i in range(h.shape[0]):
        for j in range(h.shape[1]):
            # 去除黑色像素点
            if 0 <= h[i, j] <= 180 and 0 <= v[i, j] <= 46:
                h[i, j] = 0
                s[i, j] = 0
                v[i, j] = 0
            # 去除白色像素点
            elif 0 <= h[i, j] <= 180 and 221 <= v[i, j] <= 255 and 0 <= s[i, j] <= 30:
                h[i, j] = 0
                s[i, j] = 0
                v[i, j] = 0
            # 去除 logo 区域
            if i <= 70 and 180 <= v[i, j] <= 255:
                h[i, j] = 0
                s[i, j] = 0
                v[i, j] = 0
    return [h, s, v]

# 统计各化学元素的数目
def count_points(img: list):
    h, s, v = img
    cnt = 0
    for i in range(v.shape[0]):
        for j in range(v.shape[1]):
            if v[i, j] != 0:
                cnt += 1
    return cnt

images = [Al, Fe, P]
# plot_3image(images)

Al_hsv = rgb2hsv(Al)
Fe_hsv = rgb2hsv(Fe)
P_hsv = rgb2hsv(P)
# plot_hsv_image(rgb2hsv(Al))
```

```
79
80  # 预处理，去除 logo 以及 50nm 标识
81  pre_Al = remove_LogoAndPoints(Al_hsv)
82  pre_Fe = remove_LogoAndPoints(Fe_hsv)
83  pre_P = remove_LogoAndPoints(P_hsv)
84
85  # 统计各化学元素的数目
86  cnt_Al = count_points(pre_Al)
87  cnt_Fe = count_points(pre_Fe)
88  cnt_P = count_points(pre_P)
89
90  print("铝元素的点数为: ", cnt_Al)
91  print("铁元素的点数为: ", cnt_Fe)
92  print("磷元素的点数为: ", cnt_P)
93
94  # 求解重叠量，并将元素两两的重叠量画出来
95  def cnt2overlap(img1: list, img2: list):
96      h1, s1, v1 = img1
97      h2, s2, v2 = img2
98      # 找到两者重叠的区域
99      overlap_mask = (v1 != 0) & (v2 != 0)
100     # 计算重叠量
101     cnt = np.sum(overlap_mask)
102     # 计算 h0, s0, v0
103     h0 = np.zeros_like(h1, dtype=np.uint8)
104     s0 = np.zeros_like(s1, dtype=np.uint8)
105     v0 = np.zeros_like(v1, dtype=np.uint8)
106     h0[overlap_mask] = np.clip(h1[overlap_mask] + h2[overlap_mask], 0, 255)
107     s0[overlap_mask] = np.clip(s1[overlap_mask] + s2[overlap_mask], 0, 255)
108     v0[overlap_mask] = np.clip(v1[overlap_mask] + v2[overlap_mask], 0, 255)
109     return cnt, [h0, s0, v0]
110
111 # 同理，求三者的重叠量
112 def cnt3overlap(img1: list, img2: list, img3: list):
113     h1, s1, v1 = img1
114     h2, s2, v2 = img2
115     h3, s3, v3 = img3
116     # 找到三者重叠的区域
117     overlap_mask = (v1 != 0) & (v2 != 0) & (v3 != 0)
118     # 计算重叠量
119     cnt = np.sum(overlap_mask)
120     # 计算 h0, s0, v0
121     h0 = np.zeros_like(h1, dtype=np.uint8)
122     s0 = np.zeros_like(s1, dtype=np.uint8)
123     v0 = np.zeros_like(v1, dtype=np.uint8)
124     h0[overlap_mask] = np.clip(h1[overlap_mask] + h2[overlap_mask] + h3[overlap_mask
        ], 0, 255)
```

```
125     s0[overlap_mask] = np.clip(s1[overlap_mask] + s2[overlap_mask] + s3[overlap_mask
            ], 0, 255)
126     v0[overlap_mask] = np.clip(v1[overlap_mask] + v2[overlap_mask] + v3[overlap_mask
            ], 0, 255)
127     return cnt, [h0, s0, v0]
128 
129 # 算两两重叠率
130 cnt_Al_and_Fe, img_Al_and_Fe = cnt2overlap(pre_Al, pre_Fe)
131 cnt_Al_and_P, img_Al_and_P = cnt2overlap(pre_Al, pre_P)
132 cnt_Fe_and_P, img_Fe_and_P = cnt2overlap(pre_Fe, pre_P)
133 
134 img_Al_and_Fe = cv2.cvtColor(cv2.merge(img_Al_and_Fe), cv2.COLOR_HSV2BGR)
135 img_Al_and_P = cv2.cvtColor(cv2.merge(img_Al_and_P), cv2.COLOR_HSV2BGR)
136 img_Fe_and_P = cv2.cvtColor(cv2.merge(img_Fe_and_P), cv2.COLOR_HSV2BGR)
137 
138 print("铝和铁的重叠量为: ", cnt_Al_and_Fe)
139 print("铝和磷的重叠量为: ", cnt_Al_and_P)
140 print("铁和磷的重叠量为: ", cnt_Fe_and_P)
141 # plot_3image([img_Al_and_Fe, img_Al_and_P, img_Fe_and_P])
142 
143 # 算三者重叠率
144 cnt_all, img_all = cnt3overlap(pre_Al, pre_Fe, pre_P)
145 img_all = cv2.cvtColor(cv2.merge(img_all), cv2.COLOR_HSV2BGR)
146 print("铝、铁、磷三者的重叠量为: ", cnt_all)
147 cv2.imshow('All overlap', img_all)
148 cv2.waitKey(0)
149 cv2.destroyAllWindows()
150 
151 # 保存两两元素重叠的图片、保存三者元素重叠的图片
152 if not os.path.exists("image_output"):
153     os.makedirs("image_output")
154 cv2.imwrite(os.path.join("image_output", "Al_and_Fe.jpg"), img_Al_and_Fe)
155 cv2.imwrite(os.path.join("image_output", "Al_and_P.jpg"), img_Al_and_P)
156 cv2.imwrite(os.path.join("image_output", "Fe_and_P.jpg"), img_Fe_and_P)
157 cv2.imwrite(os.path.join("image_output", "All_overlap.jpg"), img_all)
```