Q1-1.py

```
import cv2
image = cv2.imread('Figure1.jpg')

# 2. 转换为灰度图像
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# 3. 二值化处理
threshold_value = 128  # 可以根据需要调整阈值
_, binary_image = cv2.threshold(gray_image, threshold_value, 255, cv2.
                                        THRESH_BINARY)

# 4.转换为BGR图像
imgBGR = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)


# 5. 显示或保存图像
cv2.imshow('RGB Image', image)
cv2.imshow('Grayscale Image', gray_image)
cv2.imshow('Binary Image', binary_image)
cv2.imshow('BGR Image', imgBGR)

cv2.imwrite('gray_image.jpg', gray_image)
cv2.imwrite('binary_image.jpg', binary_image)
cv2.imwrite('imgBGR.jpg', imgBGR)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Q1-2.py

```
import cv2
import matplotlib.pyplot as plt

# 1. 读取彩色图像
image = cv2.imread('Figure1.jpg')

# 2. 将彩色图像从BGR颜色空间转换为HSV颜色空间
hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# 3. 对亮度（V通道）进行直方图均衡化
hsv_image[:, :, 2] = cv2.equalizeHist(hsv_image[:, :, 2])

# 4. 将处理后的图像从HSV颜色空间转换回BGR颜色空间
equalized_image = cv2.cvtColor(hsv_image, cv2.COLOR_HSV2BGR)

# 5. 绘制直方图均衡化前后的彩色图像
```

```
plt.figure(figsize=(12, 6))

# 原始彩色图像
plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title('Original Image')

# 均衡化后的彩色图像
plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(equalized_image, cv2.COLOR_BGR2RGB))
plt.title('Equalized Image')

plt.show()
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Q1-3.py

```
import cv2
import numpy as np

image = cv2.imread('Figure1.jpg')

hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# 3. 定义深红色的HSV值的范围
# 红色的范围通常为约[0, 20] 和 [160, 180]
lower_red1 = np.array([0, 45, 45])
upper_red1 = np.array([7, 255, 255])
lower_red2 = np.array([160, 45, 45])
upper_red2 = np.array([180, 255, 255])

# 创建深红色区域的掩码1和掩码2
mask1 = cv2.inRange(hsv_image, lower_red1, upper_red1)
mask2 = cv2.inRange(hsv_image, lower_red2, upper_red2)

# 合并掩码1和掩码2
mask = cv2.bitwise_or(mask1, mask2)
#将掩码应用于原始图像，获取深红色部分
red_part = cv2.bitwise_and(image, image, mask=mask)
cv2.imshow("nn",red_part)

blue_color = (240,40,0)  # 深蓝色
red_part[np.where((red_part!=[0,0,0]).all(axis=2))] = blue_color
cv2.imshow('change',red_part)

#合并深红色替换后的图像与原图像的其他部分
```

```
result_image = cv2.addWeighted(image, 1, red_part, 1, 0)

cv2.imshow('Result Image',result_image)
cv2.imwrite('result_image.jpg', result_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Q2-1.py

```
import cv2
import numpy as np

image = cv2.imread('Figure2.jpg')

# 添加椒盐噪声
def add_salt_and_pepper_noise(image, salt_prob, pepper_prob):
    noisy_image = np.copy(image)
    total_pixels = image.size

    # 添加盐噪声
    num_salt = int(total_pixels * salt_prob)
    salt_coords = [np.random.randint(0, i - 1, num_salt) for i in image.shape]
    noisy_image[salt_coords[0], salt_coords[1]] = 255

    # 添加椒噪声
    num_pepper = int(total_pixels * pepper_prob)
    pepper_coords = [np.random.randint(0, i - 1, num_pepper) for i in image.
                                            shape]
    noisy_image[pepper_coords[0], pepper_coords[1]] = 0

    return noisy_image

salt_prob = 0.01  # 设置椒盐噪声的概率
pepper_prob = 0.01

noisy_image = add_salt_and_pepper_noise(image, salt_prob, pepper_prob)

# 3. 显示或保存带噪声的图像
cv2.imshow('Noisy Image', noisy_image)
cv2.imwrite('noisy_image.jpg', noisy_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Q2-2.py

```python
import cv2

noisy_image = cv2.imread('noisy_image.jpg', cv2.IMREAD_GRAYSCALE)

# 均值滤波
denoised_mean_3x3 = cv2.blur(noisy_image, (3, 3))
denoised_mean_5x5 = cv2.blur(noisy_image, (5, 5))
denoised_mean_7x7 = cv2.blur(noisy_image, (7, 7))
denoised_mean_9x9 = cv2.blur(noisy_image, (9, 9))

# 高斯滤波
denoised_gaussian_3x3 = cv2.GaussianBlur(noisy_image, (3, 3), 0)
denoised_gaussian_5x5 = cv2.GaussianBlur(noisy_image, (5, 5), 0)
denoised_gaussian_7x7 = cv2.GaussianBlur(noisy_image, (7, 7), 0)
denoised_gaussian_9x9 = cv2.GaussianBlur(noisy_image, (9, 9), 0)

#中值滤波
denoised_median_3x3 = cv2.medianBlur(noisy_image, 3)
denoised_median_5x5 = cv2.medianBlur(noisy_image, 5)
denoised_median_7x7 = cv2.medianBlur(noisy_image, 7)
denoised_median_9x9 = cv2.medianBlur(noisy_image, 9)

# 4. 显示或保存降噪后的图像
cv2.imwrite('mean 3x3.jpg', denoised_mean_3x3)
cv2.imwrite('mean 5x5.jpg', denoised_mean_5x5)
cv2.imwrite('mean 7x7.jpg', denoised_mean_7x7)
cv2.imwrite('mean 9x9.jpg', denoised_mean_9x9)
cv2.imwrite('Gaussian 3x3.jpg', denoised_gaussian_3x3)
cv2.imwrite('Gaussian 5x5.jpg', denoised_gaussian_5x5)
cv2.imwrite('Gaussian 7x7.jpg', denoised_gaussian_7x7)
cv2.imwrite('Gaussian 9x9.jpg', denoised_gaussian_9x9)
cv2.imwrite('Median 3x3.jpg', denoised_median_3x3)
cv2.imwrite('Median 5x5.jpg', denoised_median_5x5)
cv2.imwrite('Median 7x7.jpg', denoised_median_7x7)
cv2.imwrite('Median 9x9.jpg', denoised_median_9x9)

cv2.waitKey(0)
cv2.destroyAllWindows()
```