数字图像处理实验报告

班级：通信-22 班

姓名：祝玉宪

学号：2022322010130

目录

实验项目三：频率域滤波

前言

滤波器：抑制或最小化某些频率的波或振荡的装置或材料
频率：自变量单位变化期间，一个周期函数重复相同值序列的次数

1 实现图像的傅里叶变换，显示其幅度谱的图像

1.1 实验原理

M×N 图像的而为离散傅里叶变换的公式为

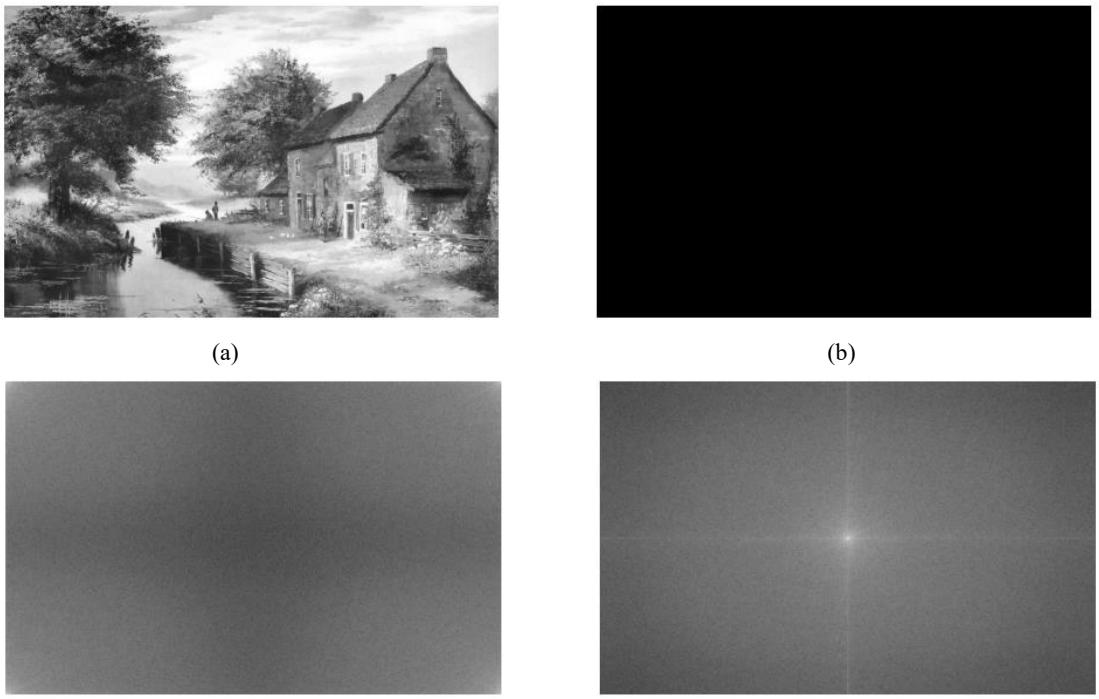
$$F(u,v)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)e^{-2\pi(\frac{ux}{M}+\frac{vy}{N})} \tag{1.1}$$

但在实际的实现中，如果自己编写相关的代码，速度是很慢的。可以调用 NumPy 自带的快速傅里叶变换的方法来显示图像的幅度谱。由于经过傅里叶变换后的图像像素值比较大，所以还需要用对数函数来使之归到 0-255 之间。

1.2 实验代码

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
img_path = "E:\桌面\作业专用图.jpg"

src = np.array(Image.open(img_path).convert("L"))

dft_src = np.fft.fft2(src)
log_src = np.log(1 + np.abs(dft_src))
ctr_src = np.fft.fftshift(log_src)
img_list = [src, dft_src, log_src, ctr_src]
img_list_name = ["原图", "DFT 图", "DFT+对数变换图", "中心化图"]

plt.rcParams['font.family'] = ['sans-serif']
plt.rcParams['font.sans-serif'] = ['SimHei']
_, img_xy = plt.subplots(2, 2, figsize=(12, 12))

for i in range(2):
    for j in range(2):
        img_xy[i][j].imshow(np.abs(img_list[i * 2 + j]), cmap="gray")
        img_xy[i][j].set_title(img_list_name[i * 2 + j], size=20)
        img_xy[i][j].axis("off")

plt.show()
```

1.3 实验结果与分析



(a) (b)

(c) (d)

图 1-1 图像傅里叶变换及其频谱图

(a) 原图 (b) DFT 图 (c) DFT+对数变换图 (d) 中心化图

可见，在频谱中心化后的图像中，中间是低频，四周都是高频；未经过频谱中心化的图像则相反。

2 用理想低通滤波器在频率域实现低通滤波

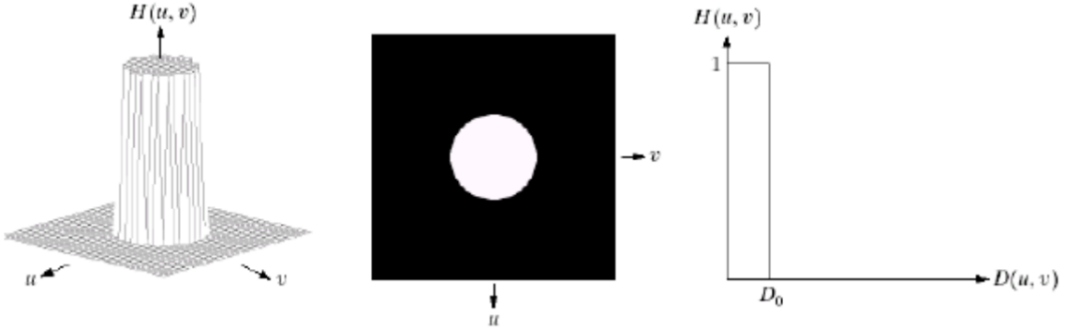
2.1 实验原理

理想低通滤波器（ILPF）的滤波器传递函数为

$$H(u,v)=\begin{cases}1 & D(u,v)\le D_0\\ 0 & D(u,v)>D_0\end{cases} \tag{2.1}$$

$$D(u,v)=[(u-M/2)^2+(v-N/2)^2]^{1/2} \tag{2.2}$$

将理想低通滤波器其用不同的方式可视化表达出来



(a) (b) (c)

图 2-1 理想低通滤波器的各种表示方式

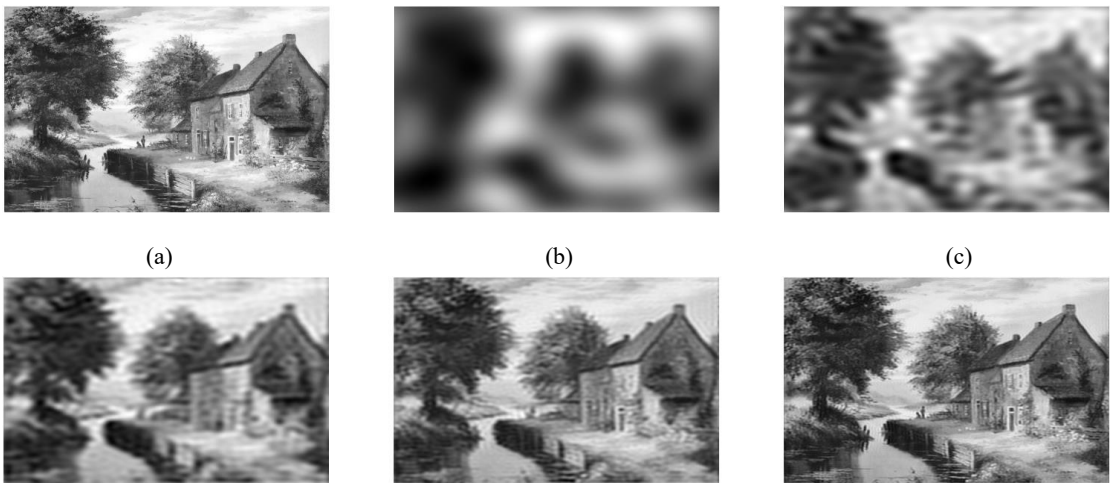
实现的方式：
（1）对图像做傅里叶变换并使得频谱中心化
（2）设定滤波的半径，并以图像中心为圆心画圆
（3）圆内的滤波器值设为 1，圆外为 0
（4）逆中心化和逆傅里叶变换

2.2 实验代码

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

def fourier_trans(src):
    """
    fourier transformation
        source: source image
        ctr_src: spectrum after fourier transformation and centralization
        plt_src: spectrum after log transformation
    """
    fft_src = np.fft.fft2(src)
    ctr_src = np.fft.fftshift(fft_src)
    plt_src = np.log(np.abs(ctr_src))

    return ctr_src, plt_src


def inv_fourier_trans(src):
    """
    inverse fourier transformation
        src: spectrums
        ifft_img: the image after inverse fourier transformation
    """
    inv_ctr_img = np.fft.ifftshift(src)
    ifft_img = np.fft.ifft2(inv_ctr_img)

    return ifft_img

def idea_low_pass_filter(source, radius=5):
    """
    LPF
        source: source image
        radius: size for low pass filter

    """
    # get the paras
```

```
        filter_radius = radius
        img = source
        # set paras for filter
        height, weight = img.shape
        center_h = int(height / 2)
        center_w = int(weight / 2)

        # initialize filter
        low_pass_filter = np.zeros_like(img)

        # set the low pass area
        for i in range(height):
            for j in range(weight):
                dist_from_center = np.sqrt(np.power((i - center_h), 2) +
np.power((j - center_w), 2))
                if dist_from_center < radius:
                    low_pass_filter[i][j] = 1

        # filter the image
        filtered_img = np.multiply(img, low_pass_filter)

        return filtered_img


img_path = "E:\桌面\作业专用图.jpg"
src = np.array(Image.open(img_path).convert("L"))
fft_src, _ = fourier_trans(src)

img_list = [src]
radius_list = ['原图', 5, 15, 30, 50, 100]
for i in radius_list[1:]:
    img_list.append(inv_fourier_trans(idea_low_pass_filter(fft_src, i)))

img_list_name = radius_list

_, img_xy = plt.subplots(2, 3, figsize=(12, 12))

for i in range(2):
    for j in range(3):
        img_xy[i][j].imshow(np.abs(img_list[i * 3 + j]), cmap="gray")
        img_xy[i][j].set_title(img_list_name[i * 3 + j], size=20)
        img_xy[i][j].axis("off")

plt.show()
```

2.3 实验结果与分析



(a) (b) (c)

(d) (e) (f)

图 2-2 不同滤波半径下滤波得到的结果图

(a) 原图 (b) 5 (c) 15 (d) 30 (e) 50 (f)100

图中总共设置了 5 种滤波的半径，可以观察到，低通滤波一是模糊，二是会有振铃效应，出现振铃效应的主要原因是理想滤波器在频域是一个门函数，但是在时域对应的是一个 sinc 函数，而频域的相乘就是时域的卷积。

时域卷积 sinc 函数就是对该函数进行搬移与叠加，那自然就有一圈圈波纹产生，也就是我们所谓的振铃效应。

3 用理想高通滤波器在频率域实现低通滤波

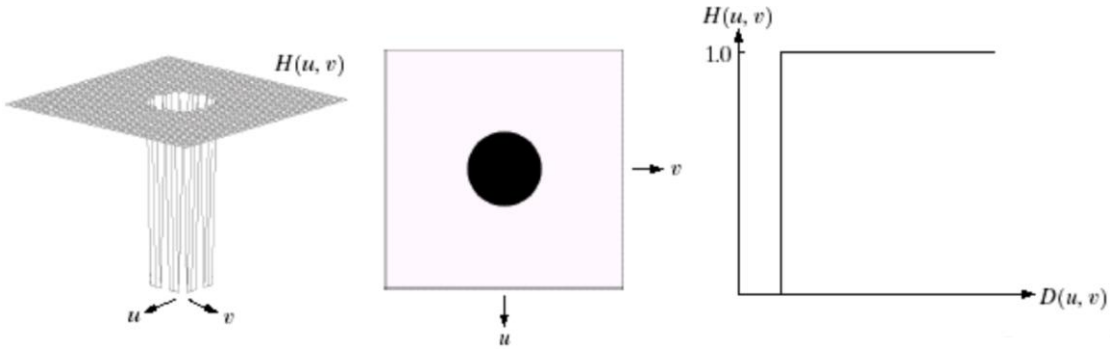
3.1 实验原理

理想高通滤波器（IHPF）滤波器传递函数为

$$H(u,v)=\begin{cases}0 & D(u,v)\le D_0\\ 1 & D(u,v)>D_0\end{cases} \tag{3.1}$$

$$D(u,v)=[(u-M/2)^2+(v-N/2)^2]^{1/2} \tag{3.2}$$

将理想高通滤波器用不同的方式可视化表达出来



(a) (b) (c)

图 3-1 理想高通滤波器的各种表示方式

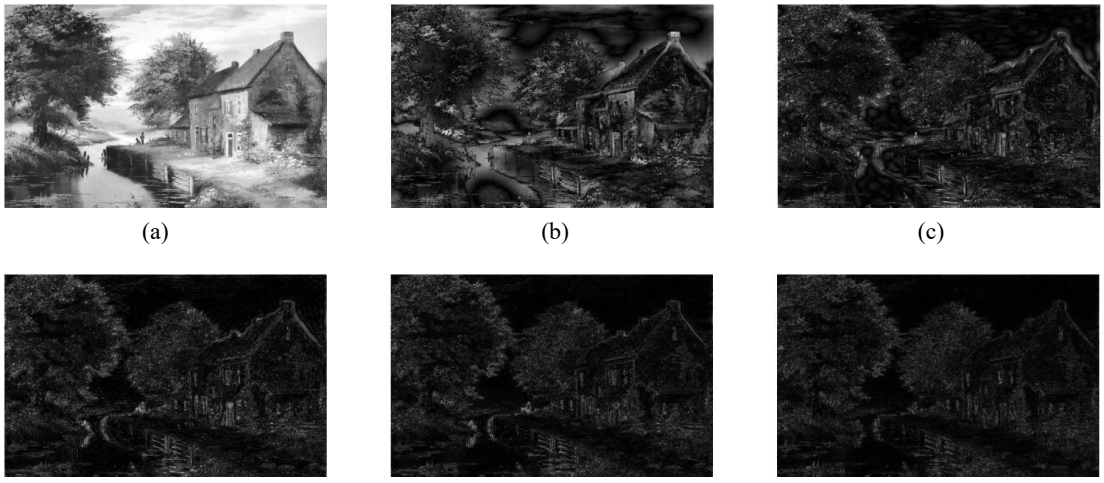
实现的方式：
（1）对图像做傅里叶变换并使得频谱中心化
（2）设定滤波的半径，并以图像中心为圆心画圆
（3）圆内的滤波器值设为 0，圆外为 1
（4）逆中心化和逆傅里叶变换

3.2 实验代码

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

def fourier_trans(src):
    """
    fourier transformation
        source: source image
        ctr_src: spectrum after fourier transformation and centralization
        plt_src: spectrum after log transformation
    """
    fft_src = np.fft.fft2(src)
    ctr_src = np.fft.fftshift(fft_src)
    plt_src = np.log(np.abs(ctr_src))

    return ctr_src, plt_src

def inv_fourier_trans(src):
    """
    inverse fourier transformation
        src: spectrums
        ifft_img: the image after inverse fourier transformation
    """
    inv_ctr_img = np.fft.ifftshift(src)
    ifft_img = np.fft.ifft2(inv_ctr_img)

    return ifft_img

def idea_high_pass_filter(source, radius=5):
    """
    HPF
        source: source image
        radius: size for High pass filter
    """
    # get the paras
    filter_radius = radius
    img = source
```

```
        # set paras for filter
        height, weight = img.shape
        center_h = int(height / 2)
        center_w = int(weight / 2)

        # initialize filter
        high_pass_filter = np.ones_like(img)

        # set the high pass area
        for i in range(height):
            for j in range(weight):
                dist_from_center = np.sqrt(np.power((i - center_h), 2) +
np.power((j - center_w), 2))

                if dist_from_center < radius:
                    high_pass_filter[i][j] = 0

        # filter the image
        filtered_img = np.multiply(img, high_pass_filter)

        return filtered_img


img_path = "E:\桌面\作业专用图.jpg"
src = np.array(Image.open(img_path).convert("L"))
fft_src, _ = fourier_trans(src)

img_list = [src]
radius_list = ['origin', 5, 15, 30, 50, 100]
for i in radius_list[1:]:
    img_list.append(inv_fourier_trans(idea_high_pass_filter(fft_src, i)))

img_list_name = radius_list

_, img_xy = plt.subplots(2, 3, figsize=(12, 12))

for i in range(2):
    for j in range(3):
        img_xy[i][j].imshow(np.abs(img_list[i * 3 + j]), cmap="gray")
        img_xy[i][j].set_title(img_list_name[i * 3 + j], size=20)
        img_xy[i][j].axis("off")

plt.show()
```

3.3 实验结果与分析



(a) (b) (c)

(d) (e) (f)

图 3-2 不同滤波半径下滤波得到的结果图

(a) 原图 (b) 5 (c) 15 (d) 30 (e) 50 (f)100

4 用巴特沃斯低通滤波器和高斯低通滤波器实现图像的低通滤波

4.1 巴特沃斯低通滤波器

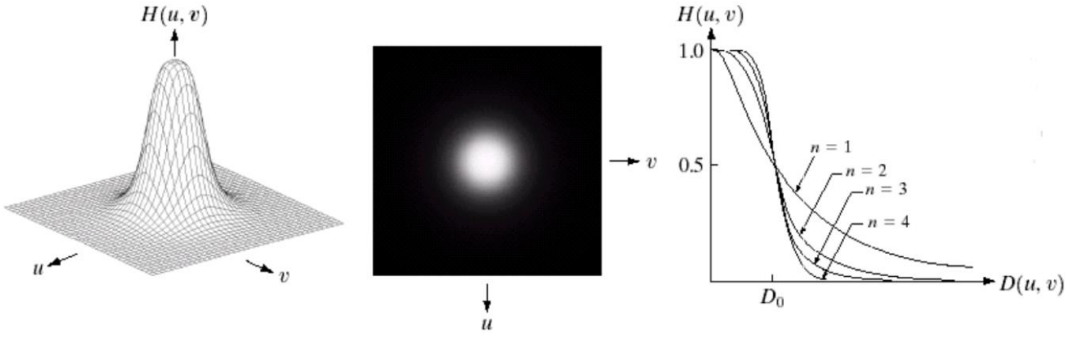
4.1 实验原理

巴特沃斯低通滤波器（BLPF）滤波器传递函数为：

$$H(u,v)=\frac{1}{1+[D(u,v)/D_0]^{2n}} \tag{4.1.1}$$

$$D(u,v)=[(u-M/2)^2+(v-N/2)^2]^{1/2} \tag{4.1.2}$$

将巴特沃斯低通滤波器用不同的方式可视化表达出来：



(a) (b) (c)

图 4-1-1 巴特沃斯低通滤波器的各种表示方式

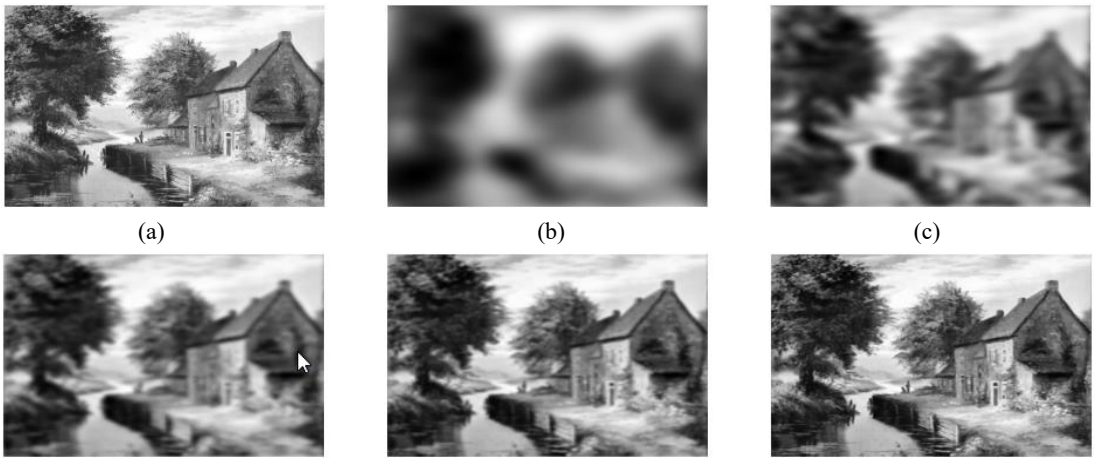
4.1.2 实验代码

```
import numpy as np
```

```
import matplotlib.pyplot as plt
from PIL import Image

def fourier_trans(src):
    """
    fourier transformation
        source: source image
        ctr_src: spectrum after fourier transformation and centralization
        plt_src: spectrum after log transformation
    """
    fft_src = np.fft.fft2(src)
    ctr_src = np.fft.fftshift(fft_src)
    plt_src = np.log(np.abs(ctr_src))

    return ctr_src, plt_src

def inv_fourier_trans(src):
    """
    inverse fourier transformation
        src: spectrums
        ifft_img: the image after inverse fourier transformation
    """
    inv_ctr_img = np.fft.ifftshift(src)
    ifft_img = np.fft.ifft2(inv_ctr_img)

    return ifft_img

def butterworth_low_pass_filter(source, radius=5, order=1):
    # get the paras
    filter_radius = radius
    img = source

    # set paras for filter
    height, weight = img.shape
    center_h = int(height / 2)
    center_w = int(weight / 2)

    # initialize filter
    butterworth_low_pass_filter = np.zeros_like(img)

    # set the pass area
    for i in range(height):
        for j in range(weight):
            dist_from_center = np.sqrt(np.power((i - center_h), 2) +
```

```
np.power((j - center_w), 2))
                butterworth_low_pass_filter[i][j] = 1 / (1 +
np.power(dist_from_center / radius, 2 * order))

        # filter the image
        filtered_img = np.multiply(img, butterworth_low_pass_filter)

        return filtered_img

    img_path = "E:\桌面\作业专用图.jpg"
    src = np.array(Image.open(img_path).convert("L"))
    fft_src, _ = fourier_trans(src)

    img_list = [src]
    radius_list = ['origin', 5, 15, 30, 50, 100]
    for i in radius_list[1:]:
        img_list.append(inv_fourier_trans(butterworth_low_pass_filter(fft_src, i, 2)))

    img_list_name = radius_list

    _, img_xy = plt.subplots(2, 3, figsize=(12, 12))
    for i in range(2):
        for j in range(3):
            img_xy[i][j].imshow(np.abs(img_list[i * 3 + j]), cmap="gray")
            img_xy[i][j].set_title(img_list_name[i * 3 + j], size=20)
            img_xy[i][j].axis("off")

    plt.show()
```

4.1.3 实验结果与分析



(a) (b) (c)

(d) (e) (f)

图 4-1-2 不同滤波半径下滤波得到的结果图

(a) 原图 (b) 5 (c) 15 (d) 30 (e) 50 (f)100

观察可以发现，巴特沃斯低通滤波器虽然依然有模糊，但基本上可以说没有振铃效应。

4.2 高斯低通滤波器

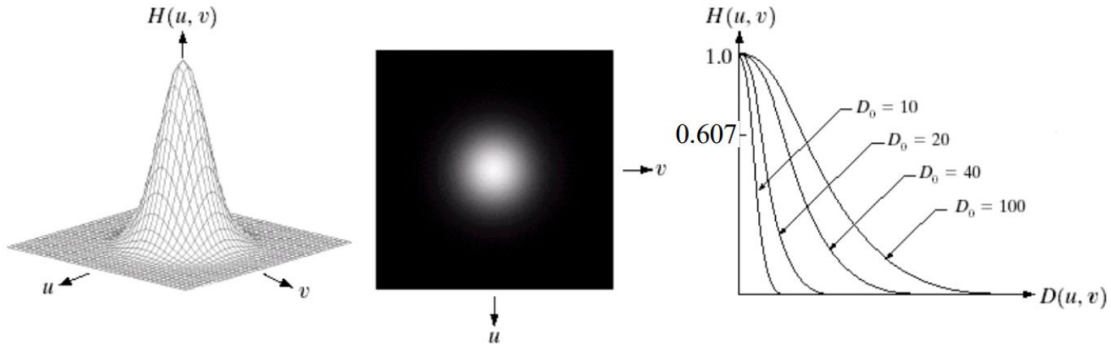
4.2.1 实验原理

高斯低通滤波器（GLPF）滤波器传递函数为：

$$H(u,v) = e^{-D^2(u,v)/2D_0^2} \tag{4.2.1}$$

$$D(u,v) = [(u - M/2)^2 + (v - N/2)^2]^{1/2} \tag{4.2.2}$$

将高斯低通滤波器其用不同的方式可视化表达出来：



(a) (b) (c)

图 4-2-1 高斯低通滤波器的各种表示方式

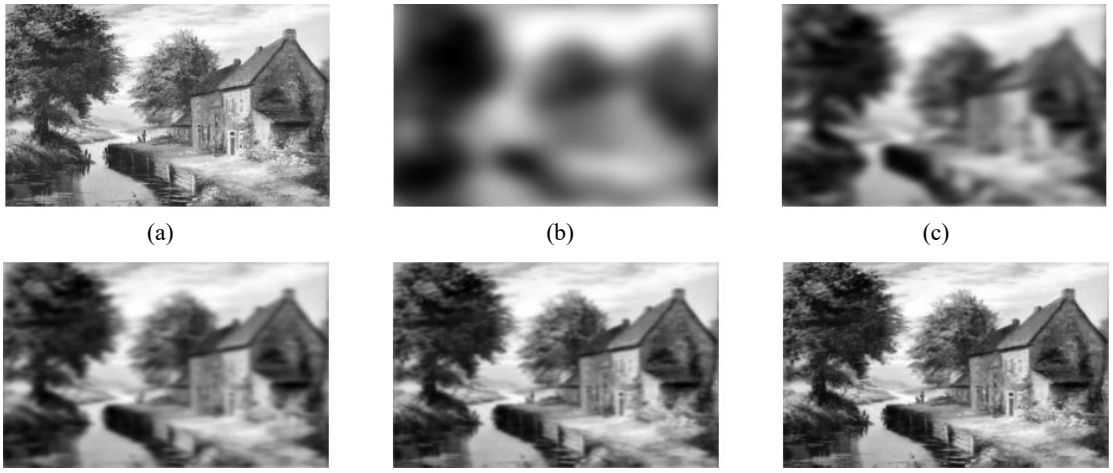
4.2.2 实验代码

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
def fourier_trans(src):
    """
    fourier transformation
        source: source image
        ctr_src: spectrum after fourier transformation and centralization
        plt_src: spectrum after log transformation
    """
    fft_src = np.fft.fft2(src)
    ctr_src = np.fft.fftshift(fft_src)
    plt_src = np.log(np.abs(ctr_src))
    return ctr_src, plt_src

def inv_fourier_trans(src):
    """
    inverse fourier transformation
        src: spectrums
```

```
            ifft_img: the image after inverse fourier transformation
        """
        inv_ctr_img = np.fft.ifftshift(src)
        ifft_img = np.fft.ifft2(inv_ctr_img)

        return ifft_img

    def guass_low_pass_filter(source, radius=5):
        # get the paras
        filter_radius = radius
        img = source

        # set paras for filter
        height, weight = img.shape
        center_h = int(height / 2)
        center_w = int(weight / 2)

        # initialize filter
        guass_low_pass_filter = np.zeros_like(img)

        # set the pass area
        for i in range(height):
            for j in range(weight):
                dist_from_center = np.sqrt(np.power((i - center_h), 2) +
np.power((j - center_w), 2))
                guass_low_pass_filter[i][j] = np.exp(-(np.power(dist_from_center,
2) / np.power(radius, 2)))
        # filter the image
        filtered_img = np.multiply(img, guass_low_pass_filter)
        return filtered_img

    img_path = "E:\桌面\作业专用图.jpg"
    src = np.array(Image.open(img_path).convert("L"))
    fft_src, _ = fourier_trans(src)
    img_list = [src]
    radius_list = ['origin', 5, 15, 30, 50, 100]
    for i in radius_list[1:]:
        img_list.append(inv_fourier_trans(guass_low_pass_filter(fft_src, i)))
    img_list_name = radius_list
    _, img_xy = plt.subplots(2, 3, figsize=(12, 12))
    for i in range(2):
        for j in range(3):
            img_xy[i][j].imshow(np.abs(img_list[i * 3 + j]), cmap="gray")
            img_xy[i][j].set_title(img_list_name[i * 3 + j], size=20)
```

```
            img_xy[i][j].axis("off")
    plt.show()
```

4.2.3 实验结果与分析



(a) (b) (c)

(d) (e) (f)

图 4-2-2 不同滤波半径下滤波得到的结果图

(a) 原图 (b) 5 (c) 15 (d) 30 (e) 50 (f)100

可以观察到，高斯低通滤波器同样是模糊但是没有振铃效应。

5 用巴特沃斯高通滤波器和高斯高通滤波器实现图像的高频增强

5.1 巴特沃斯高通滤波器

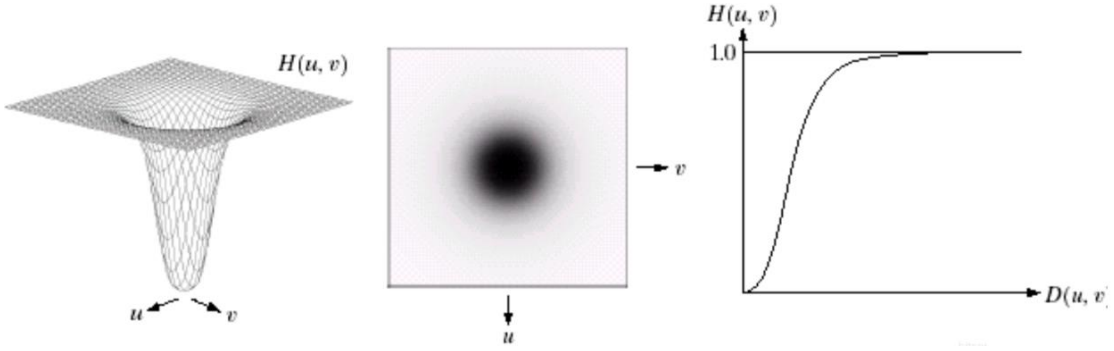
5.1.1 实验原理

巴特沃斯高通滤波器（BHPF）滤波器传递函数为：

$$H(u,v)=\frac{1}{1+[D_0/D(u,v)/]^{2n}} \tag{5.1.1}$$

$$D(u,v)=[(u-M/2)^2+(v-N/2)^2]^{1/2} \tag{5.1.2}$$

将巴特沃斯高通滤波器用不同的方式可视化表达出来：



(a) (b) (c)

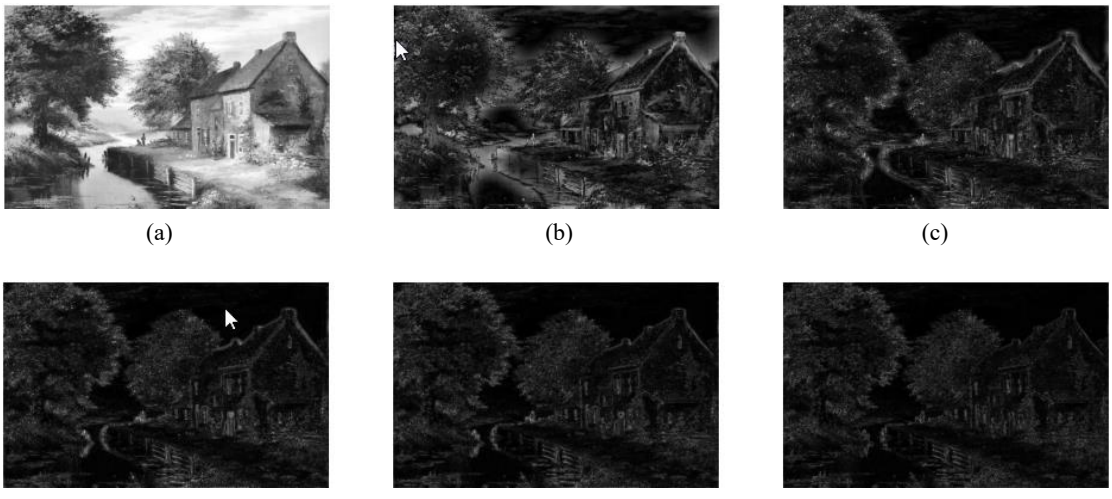
图 5-1-1 巴特沃斯高通滤波器的各种表示方式

5.1.2 实验代码

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

def fourier_trans(src):
    """
    fourier transformation
        source: source image
        ctr_src: spectrum after fourier transformation and centralization
        plt_src: spectrum after log transformation
    """
    fft_src = np.fft.fft2(src)
    ctr_src = np.fft.fftshift(fft_src)
    plt_src = np.log(np.abs(ctr_src))
    return ctr_src, plt_src

def inv_fourier_trans(src):
    """
    inverse fourier transformation
        src: spectrums
        ifft_img: the image after inverse fourier transformation
    """
    inv_ctr_img = np.fft.ifftshift(src)
    ifft_img = np.fft.ifft2(inv_ctr_img)
    return ifft_img

def butterworth_high_pass_filter(source, radius=5, order=2):
    # get the paras
    filter_radius = radius
    img = source
    # set paras for filter
    height, weight = img.shape
    center_h = int(height / 2)
    center_w = int(weight / 2)
    # initialize filter
    butterworth_high_pass_filter = np.zeros_like(img)
    # set the pass area
    for i in range(height):
        for j in range(weight):
            dist_from_center = np.sqrt(np.power((i - center_h), 2) +
np.power((j - center_w), 2))
            if dist_from_center != 0:
```

```
                butterworth_high_pass_filter[i][j] = 1 / (1 + np.power(radius /
dist_from_center, 2 * order))
        # filter the image
        filtered_img = np.multiply(img, butterworth_high_pass_filter)

        return filtered_img

img_path = "E:\桌面\作业专用图.jpg"
src = np.array(Image.open(img_path).convert("L"))
fft_src, _ = fourier_trans(src)

img_list = [src]
radius_list = ['origin', 5, 15, 30, 50, 100]
for i in radius_list[1:]:
    img_list.append(inv_fourier_trans(butterworth_high_pass_filter(fft_src, i)))

img_list_name = radius_list

_, img_xy = plt.subplots(2, 3, figsize=(12, 12))

for i in range(2):
    for j in range(3):
        img_xy[i][j].imshow(np.abs(img_list[i * 3 + j]), cmap="gray")
        img_xy[i][j].set_title(img_list_name[i * 3 + j], size=20)
        img_xy[i][j].axis("off")

plt.show()
```

5.1.3 实验结果与分析



(a) (b) (c)

(d) (e) (f)

图 5-1-2 不同滤波半径下滤波得到的结果图

(a) 原图 (b) 5 (c) 15 (d) 30 (e) 50 (f)100

5.2 高斯高通滤波器

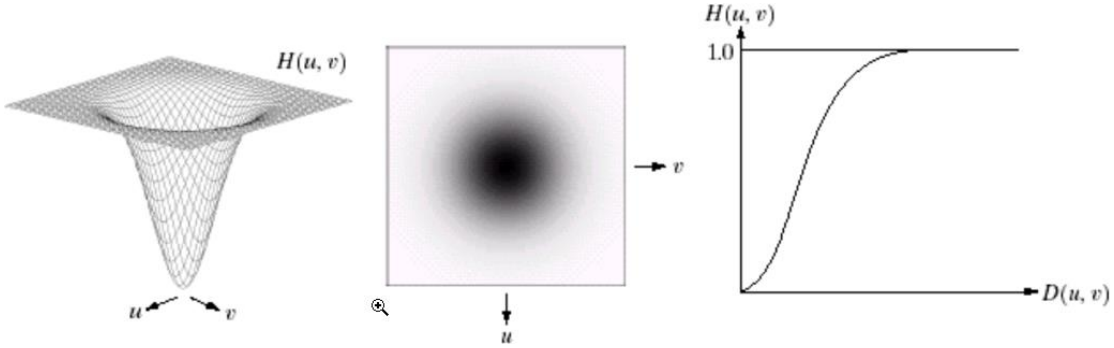
5.2.1 实验原理

高斯高通滤波器（GHPF）滤波器传递函数为：

$$H(u,v) = 1 - e^{-D^2(u,v)/2D_0^2} \tag{5.2.1}$$

$$D(u,v) = [(u - M/2)^2 + (v - N/2)^2]^{1/2} \tag{5.2.2}$$

将高斯高通滤波器用不同的方式可视化表达出来：



(a) (b) (c)

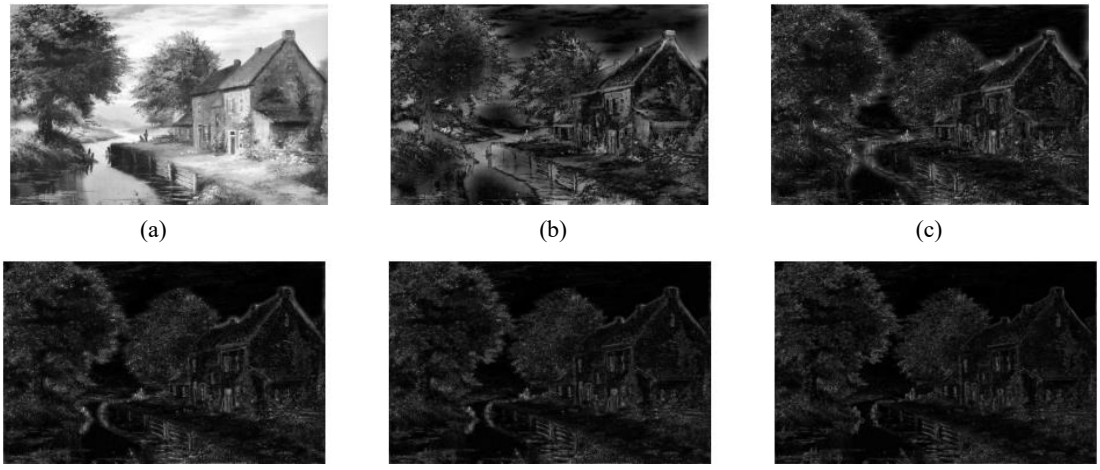
图 5-2-1 巴特沃斯高通滤波器的各种表示方式

5.2.2 实验代码

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

def fourier_trans(src):
    """
    fourier transformation
        source: source image
        ctr_src: spectrum after fourier transformation and centralization
        plt_src: spectrum after log transformation
    """
    fft_src = np.fft.fft2(src)
    ctr_src = np.fft.fftshift(fft_src)
    plt_src = np.log(np.abs(ctr_src))
    return ctr_src, plt_src

def inv_fourier_trans(src):
    """
    inverse fourier transformation
        src: spectrums
        ifft_img: the image after inverse fourier transformation
```

```
        """
        inv_ctr_img = np.fft.ifftshift(src)
        ifft_img = np.fft.ifft2(inv_ctr_img)
        return ifft_img

    def guass_high_pass_filter(source, radius=5):
        # get the paras
        filter_radius = radius
        img = source
        # set paras for filter
        height, weight = img.shape
        center_h = int(height / 2)
        center_w = int(weight / 2)
        # initialize filter
        guass_high_pass_filter = np.zeros_like(img)
        # set the pass area
        for i in range(height):
            for j in range(weight):
                dist_from_center = np.sqrt(np.power((i - center_h), 2) +
np.power((j - center_w), 2))
                guass_high_pass_filter[i][j] = 1 - np.exp(-
(np.power(dist_from_center, 2) / np.power(radius, 2)))
        # filter the image
        filtered_img = np.multiply(img, guass_high_pass_filter)
        return filtered_img

    img_path = "E:\桌面\作业专用图.jpg"
    src = np.array(Image.open(img_path).convert("L"))
    fft_src, _ = fourier_trans(src)
    img_list = [src]
    radius_list = ['origin', 5, 15, 30, 50, 100]
    for i in radius_list[1:]:
        img_list.append(inv_fourier_trans(guass_high_pass_filter(fft_src, i)))

    img_list_name = radius_list
    _, img_xy = plt.subplots(2, 3, figsize=(12, 12))

    for i in range(2):
        for j in range(3):
            img_xy[i][j].imshow(np.abs(img_list[i * 3 + j]), cmap="gray")
            img_xy[i][j].set_title(img_list_name[i * 3 + j], size=20)
            img_xy[i][j].axis("off")

    plt.show()
```

5.2.3 实验结果与分析



(a) (b) (c)

(d) (e) (f)

图 5-2-2 不同滤波半径下滤波得到的结果图

(a) 原图 (b) 5 (c) 15 (d) 30 (e) 50 (f)100