**数字图像处理第五次作业**

浮焕然 PB22061345

问题：

1. 写出逆滤波和维纳滤波图象恢复的具体步骤。

逆滤波：

* 1. 对和h做傅里叶变换得到和H。
  2. 设计恢复转移函数M：
  3. 对和M做卷积运算得到=
  4. 对做逆傅里叶变换得到

维纳滤波：

1. 对和h做傅里叶变换得到和H。
2. 设计恢复转移函数M：
3. 对和M做卷积运算得到=
4. 对做逆傅里叶变换得到
5. 推导水平匀速直线运动模糊的点扩展函数的数学公式并画出曲线。

实际采集的运动模糊图像为：

傅里叶变换为：

点扩展函数为：

对于水平匀速直线运动

则：

已知傅里叶变换：

由平移性质：

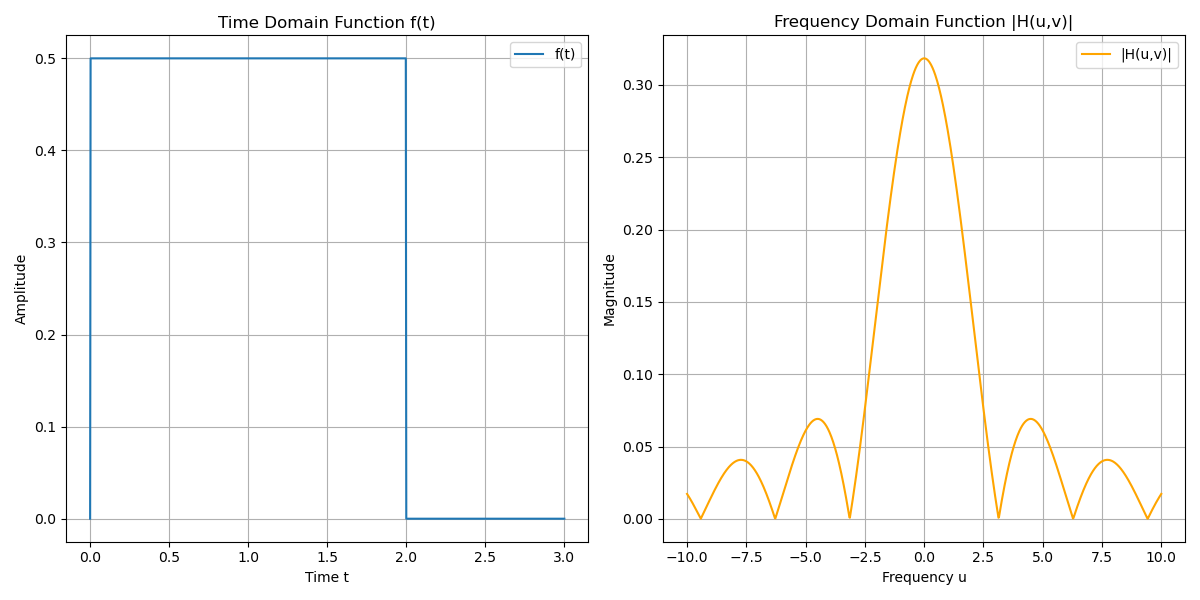


图 1.点扩展曲线图像

1. 编程实现lema.bmp的任意角旋转。

正向映射公式：

反向映射公式：

旋转步骤：

1. 计算新的图像维度，初始化为0
2. 对每一个点利用逆映射公式计算在原图中的对应点
3. 使用双线性插值（或最近邻或其他方法）计算当前点的灰度值

程序设计：

1. 使用双线性插值
2. 以中心点作为旋转点

可视化结果展示：



图 2.旋转20°



图 3.旋转70°

第3题代码如下：

import **cv2**

import **numpy** as **np**

import **math**

def **rotate\_image**(image, angle):

    """

    旋转图像

    :param image: 输入图像 (numpy数组)

    :param angle: 旋转角度 (度)

    :return: 旋转后的图像

    """

*# 获取图像的尺寸*

    height, width = image.shape[:2]

*# 计算旋转中心（图像中心）*

    center\_x, center\_y = width // 2, height // 2

*# 将角度转换为弧度*

    angle\_rad = **math**.**radians**(angle)

*# 计算旋转矩阵*

    cos\_theta = **math**.**cos**(angle\_rad)

    sin\_theta = **math**.**sin**(angle\_rad)

*# 计算新图像的边界*

    new\_width = **int**(**abs**(width \* cos\_theta) + **abs**(height \* sin\_theta))

    new\_height = **int**(**abs**(width \* sin\_theta) + **abs**(height \* cos\_theta))

*# 创建一个新的图像数组，初始化为0（黑色）*

    rotated\_image = **np**.**zeros**((new\_height, new\_width, 3), dtype=**np**.uint8)

*# 遍历新图像的每个像素，计算其在原图像中的位置*

    for y in **range**(new\_height):

        for x in **range**(new\_width):

*# 将新图像的坐标平移到以旋转中心为原点*

            x\_rel = x - new\_width // 2

            y\_rel = y - new\_height // 2

*# 应用逆旋转公式*

            x\_orig\_rel = **int**(x\_rel \* cos\_theta + y\_rel \* sin\_theta)

            y\_orig\_rel = **int**(-x\_rel \* sin\_theta + y\_rel \* cos\_theta)

*# 将原图像的坐标平移回原坐标系*

            x\_orig = x\_orig\_rel + center\_x

            y\_orig = y\_orig\_rel + center\_y

*# 检查是否在原图像的边界内*

            if 0 <= x\_orig < width and 0 <= y\_orig < height:

*# 使用双线性插值计算新像素值*

                rotated\_image[y, x] = **bilinear\_interpolation**(image, x\_orig, y\_orig)

    return rotated\_image

def **bilinear\_interpolation**(image, x, y):

    """

    双线性插值

    :param image: 输入图像

    :param x: 目标像素的x坐标

    :param y: 目标像素的y坐标

    :return: 插值后的像素值

    """

*# 获取四个最近的整数坐标*

    x1, y1 = **int**(x), **int**(y)

    x2, y2 = **min**(x1 + 1, image.shape[1] - 1), **min**(y1 + 1, image.shape[0] - 1)

*# 获取四个点的像素值*

    Ia = image[y1, x1]

    Ib = image[y1, x2]

    Ic = image[y2, x1]

    Id = image[y2, x2]

*# 计算插值*

    u = x - x1

    v = y - y1

    return (1 - u) \* (1 - v) \* Ia + u \* (1 - v) \* Ib + (1 - u) \* v \* Ic + u \* v \* Id

*# 读取图像*

image = **cv2**.**imread**('lena.bmp')

*# 指定旋转角度*

angle = 20  *# 旋转角度*

*# 旋转图像*

rotated\_image = **rotate\_image**(image, angle)

*# 显示结果*

**cv2**.**imshow**('Original Image', image)

**cv2**.**imshow**('Rotated Image', rotated\_image)

**cv2**.**waitKey**(0)

**cv2**.**destroyAllWindows**()

