# 数字图象处理

# 课程作业三

2019010485　自 91　刘祖炎*

2021 年 10 月 17 日

## 1　题目一

### *1.1*　算法原理

#### 1.1.1　生成脉冲函数

生成脉冲函数的原理较为简单，输入脉冲坐标 $xCoor$、$yCoor$，令对应位置函数值为 1 即可，代码如下所示。

```
function results = GenPulse(~, xCoor, yCoor)
    results = zeros(256, 256);
    results(xCoor, yCoor) = 1;
end
```

#### 1.1.2　生成二维正弦函数

二维正弦函数的公式为：

$$f(x,y) = \sin(2\pi f(x\cos\theta + y\sin\theta) + \phi) \tag{1}$$

函数输入正弦函数的参数：频率 $f$、角度 $\theta$、相位 $\phi$，按照公式 (1) 进行计算，输出正弦函数图像，代码如下所示。

```
function results = GenSin(~, Freq, Angle, Phase)
    [Xaxis, Yaxis] = meshgrid(1:256);
    Angle = Angle * pi / 180;
    Phase = Phase * pi / 180;
    results = sin(2 * pi * Freq * (Xaxis * cos(Angle) + Yaxis * sin(Angle)) + Phase);
end
```

其中，*meshgrid* 函数用于生成与坐标对应的二维数组用于进行计算，在运用公式前，需要将 $\theta$、$\phi$ 由角度值转化为弧度制。

*liuzuyan19@mails.tsinghua.edu.cn

### 1.1.3 生成矩形函数

函数输入矩形函数的参数：中心点坐标 $xCoor$、$yCoor$、旋转角度 $\theta$、边长 $xLength$、$yLength$，输出矩形函数图像，代码如下所示。

```
function results = GenRec(~, xCoor, yCoor, Angle, xLength, yLength)
    results = zeros(256, 256);
    xStart = max(floor((xCoor - xLength / 2)), 1);
    yStart = max(floor((yCoor - yLength / 2)), 1);
    xEnd = min(floor((xCoor + xLength / 2)), 256);
    yEnd = min(floor((yCoor + yLength / 2)), 256);
    results(xStart:xEnd, yStart:yEnd) = 1;
    results = imrotate(results, Angle, "bicubic", "crop");
end
```

根据中心点坐标与边长计算矩形四角的位置 $xStart$、$xEnd$、$yStart$、$yEnd$，并利用 $min()$、$max()$ 函数限定对应坐标不发生越界。将最终结果中对应的部分赋值为 1，并通过 $imrotate$ 函数实现矩阵的按角度旋转。

### 1.1.4 生成二维高斯函数

二维高斯函数的公式为：

$$f(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2}$$

函数输入二维高斯函数的方差 $\sigma^2$，按照公式 (2) 进行计算，输出二维高斯函数图像，代码如下所示：

```
function results = GenGauss(~, SD)
    [Xaxis, Yaxis] = meshgrid(1:256);
    results = exp(-((Xaxis - 128) .* (Xaxis - 128) + (Yaxis - 128)
    .* (Yaxis - 128)) / (2 * SD));
end
```

需要注意的是，为使生成的二维高斯函数位于中心位置，计算时需要将 $x$ 轴、$y$ 轴坐标进行中心化。

### 1.1.5 生成 Gabor 函数

Gabor 函数表示为二维高斯函数与二维正弦函数相乘，其公式为：

$$f(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \cdot \sin(2\pi f(x\cos\theta + y\sin\theta) + \phi) \tag{3}$$

函数输入正弦函数的参数：频率 $f$、角度 $\theta$、相位 $\phi$ 与二维高斯函数的方差 $\sigma^2$，按照公式 (3) 进行计算，输出 Gabor 函数图像，代码如下所示：

```
function results = GenGabor(~, Freq, Angle, Phase, SD)
    [Xaxis, Yaxis] = meshgrid(1:256);
    Angle = Angle * pi / 180;
    Phase = Phase * pi / 180;
```

```
    results = sin(2 * pi * Freq * (Xaxis * cos(Angle) + Yaxis * sin(Angle)) + Phase);
    results = results .* exp(-((Xaxis - 128) .* (Xaxis - 128) + (Yaxis - 128)
     .* (Yaxis - 128)) / (2 * SD));
end
```

其代码为高斯函数与正弦函数的融合，此处不作赘述。

### 1.1.6 DFT 变换

二维 DFT 公式为：

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-2\pi j(\frac{ux}{M}+\frac{vy}{N})} \tag{4}$$

在本题条件中，满足 $M = N$。根据公式 (4)，令：

$$\omega = e^{-\frac{2\pi j}{N}}$$

构造矩阵：

$$W = \begin{bmatrix} 1 & 1 & 1 & ... & 1 \\ 1 & \omega & \omega^2 & ... & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & ... & \omega^{2(N-1)} \\ ... & ... & ... & ... & ... \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & ... & \omega^{(N-1)(N-1)} \end{bmatrix}$$

则公式 (4) 可表示为矩阵形式：

$$F_{uv} = W \cdot f_{xy} \cdot W \tag{5}$$

对应代码如下：

```
function results = DFT(~, Image)
    index_matrix = 0:1:255;
    index_matrix = index_matrix .* transpose(index_matrix);
    omega = exp(-2 * pi * 1i / 256);
    index_matrix = omega .^ index_matrix;
    results = index_matrix * Image * index_matrix;
    results = circshift(results, 128, 1);
    results = circshift(results, 128, 2);
end
```

为构造 $W$ 矩阵，首先考虑构造对应的幂指数矩阵：

$$W^* = \begin{bmatrix} 0 & 0 & 0 & ... & 0 \\ 0 & 1 & 2 & ... & N-1 \\ 0 & 2 & 4 & ... & 2(N-1) \\ ... & ... & ... & ... & ... \\ 0 & N-1 & 2(N-1) & ... & (N-1)(N-1) \end{bmatrix}$$

在代码中，生成 1 至 256 的数组 *index_matrix*，并与其自身转置相乘即可得幂指数矩阵。利用 *circshift* 函数将 DFT 后结果的中心移动至图像中心处。

### 1.1.7 归一化

归一化函数用于将对应图像的值归一化至 $[0,1]$ 区间内，其公式为：

$$f(x,y) = \frac{f(x,y) - min(f(x,y))}{max(f(x,y)) - min(f(x,y))} \tag{6}$$

对应代码如下：

```
function results = Normalize(~, Image)
    maxNum = max(max(Image));
    minNum = min(min(Image));
    results = (Image - minNum) / (maxNum - minNum);
end
```

## *1.2 UI* 界面

程序 UI 界面如图1所示。

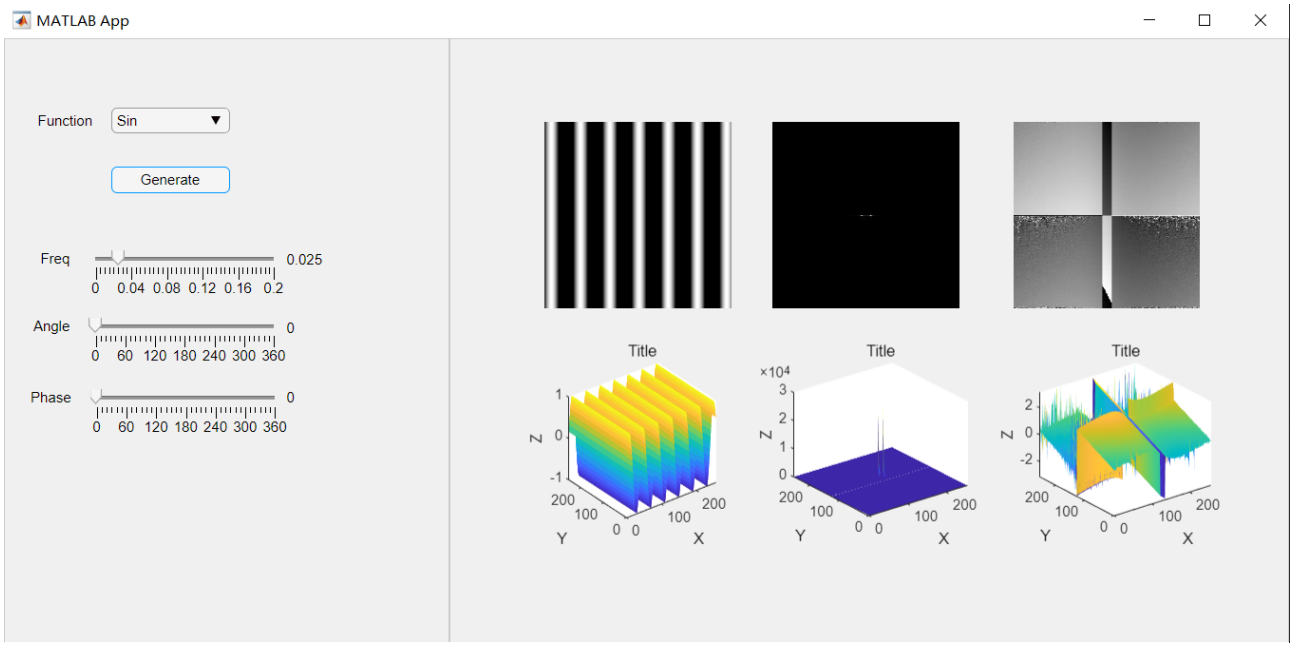


图 1: UI 界面

在左上角下拉框中选择函数类型。在左侧滑条处调节对应的参数，完成调节后点击 Generate 按钮即可生成对应的原图、幅度谱、相位谱，在每个滑条右侧会显示当前的对应值，对应的可调参数数量与标签会随着当前函数类型而发生变化。

## *1.3* 回调函数分析

### 1.3.1 滑条可见性调节函数

为在选择不同选项时显示对应数量与位置的滑条，实现滑条可见性调节函数，在改变生成函数类型时，需要输入五个滑条的可见性 (True or False) 进行控制，代码如下所示：

```
function VisibleAdjust(app, state1, state2, state3, state4, state5)
    app.Param1Slider.Visible = state1;
    app.Param1Label.Visible = state1;
    app.Param1SliderLabel.Visible = state1;
    app.Param2Slider.Visible = state2;
    app.Param2Label.Visible = state2;
    app.Param2SliderLabel.Visible = state2;
    app.Param3Slider.Visible = state3;
    app.Param3Label.Visible = state3;
    app.Param3SliderLabel.Visible = state3;
    app.Param4Slider.Visible = state4;
    app.Param4Label.Visible = state4;
    app.Param4SliderLabel.Visible = state4;
    app.Param5Slider.Visible = state5;
    app.Param5Label.Visible = state5;
    app.Param5SliderLabel.Visible = state5;
end
```

### 1.3.2 初始化函数

初始化函数如下所示：

```
function startupFcn(app)
    VisibleAdjust(app, true, true, false, false, false);
    app.Param1Slider.Limits = [1, 256];
    app.Param1Slider.Value = 128;
    app.Param1SliderLabel.Text = 'x coor';
    app.Param1Label.Text = num2str(app.Param1Slider.Value);
    app.Param2Slider.Limits = [1, 256];
    app.Param2Slider.Value = 128;
    app.Param2SliderLabel.Text = 'y coor';
    app.Param2Label.Text = num2str(app.Param2Slider.Value);
end
```

初始状态为脉冲函数状态，对应改变标签内容，设置 $x$ 轴坐标、$y$ 轴坐标位置为 $(128, 128)$。

### 1.3.3 滑条 1 回调函数

调节滑条 1 时，将对应的调节结果赋予对应的标签中以进行显示，若对应三角函数或 Gabor 函数 (频率)，则保留三位小数，其余情况保留整数。代码如下所示：

```
function Param1SliderValueChanging(app, event)
    changingValue = event.Value;
    if strcmp(app.FunctionDropDown.Value, 'Sin') ||
        strcmp(app.FunctionDropDown.Value, 'Gabor')
        app.Param1Label.Text = num2str(roundn(changingValue, -3));
    else
        app.Param1Label.Text = num2str(round(changingValue));
    end
end
```

### 1.3.4 滑条 2 回调函数

调节滑条 2 时，将对应的调节结果赋予对应的标签中以进行显示，结果保留整数。代码如下所示：

```
function Param2SliderValueChanging(app, event)
    changingValue = event.Value;
    app.Param2Label.Text = num2str(round(changingValue));
end
```

### 1.3.5 滑条 3、4、5 回调函数

滑条 3、4、5 的回调函数与滑条 2 完全相同，在此处不再赘述。

### 1.3.6 图像生成按钮回调函数

图像生成按钮回调函数如下所示：

```
function GenerateButtonPushed(app, event)
    xAxis = 1:256;
    yAxis = 1:256;
    if strcmp(app.FunctionDropDown.Value, 'Pulse')
        initImage = GenPulse(app, round(app.Param1Slider.Value),
        round(app.Param2Slider.Value));
    elseif strcmp(app.FunctionDropDown.Value, 'Sin')
        initImage = GenSin(app, app.Param1Slider.Value,
        app.Param2Slider.Value, app.Param3Slider.Value);
    elseif strcmp(app.FunctionDropDown.Value, 'Rectangle')
        initImage = GenRec(app, app.Param1Slider.Value,
        app.Param2Slider.Value, app.Param3Slider.Value,
        app.Param4Slider.Value, app.Param5Slider.Value);
    elseif strcmp(app.FunctionDropDown.Value, 'Gauss')
        initImage = GenGauss(app, app.Param1Slider.Value);
```

```
    elseif strcmp(app.FunctionDropDown.Value, 'Gabor')
        initImage = GenGabor(app, app.Param1Slider.Value,
        app.Param2Slider.Value, app.Param3Slider.Value,
        app.Param4Slider.Value);
    end

    app.Image.ImageSource = repmat(initImage, 1, 1, 3);
    s1 = surf(app.UIAxes, initImage(xAxis, yAxis));
    s1.EdgeColor = 'none';

    DFTImage = DFT(app, initImage);
    app.Image_2.ImageSource = repmat(Normalize(app, abs(DFTImage)), 1, 1, 3);
    s2 = surf(app.UIAxes_2, abs(DFTImage(xAxis, yAxis)));
    s2.EdgeColor = 'none';

    angleImage = angle(DFTImage);
    app.Image_3.ImageSource = repmat(Normalize(app, angleImage), 1, 1, 3);
    s3 = surf(app.UIAxes_3, angleImage(xAxis, yAxis));
    s3.EdgeColor = 'none';
end
```

当按下生成按钮时，根据对应下拉菜单所选函数类型生成对应的函数图片 *initImage*。利用 *repmat* 函数将该图由灰度图转化为 RGB 图供 *app* 中 *Image* 类使用。利用 *surf* 函数将三维坐标绘制结果赋予 *app* 中 *UIAxes* 类，此处可利用 *xAxis*、*yAxis* 对图片进行索引。在本作业中，为使显示精细度更高，取索引步长为 1。可通过调节 *EdgeColor* 属性隐藏三维面片的边，提升图片美观度。

完成原图生成后，调用 *DFT* 函数生成频域结果。利用灰度图进行展示时，需要首先利用 *abs*() 函数将频域复数值转化为幅度谱，并利用 *Normalize* 函数进行归一化。利用三维 *surf* 进行展示时，同样需要利用 *abs*() 函数将频域复数值转化为幅度谱。

完成幅度谱生成后，调用 *angle* 函数生成相位谱，按照相同的方式进行显示即可。

### 1.3.7 下拉菜单回调函数

下拉菜单回调函数如下所示。为简单起见，此处只展示脉冲函数的调节代码，其余代码与之思路完全相同。

```
    function FunctionDropDownValueChanged(app, event)
    value = app.FunctionDropDown.Value;
    if strcmp(value, 'Pulse')
        VisibleAdjust(app, true, true, false, false, false);
        app.Param1Slider.Limits = [1, 256];
        app.Param1Slider.Value = 128;
        app.Param1SliderLabel.Text = 'x coor';
        app.Param1Label.Text = num2str(app.Param1Slider.Value);
        app.Param2Slider.Limits = [1, 256];
        app.Param2Slider.Value = 128;
```

```
            app.Param2SliderLabel.Text = 'y coor';
            app.Param2Label.Text = num2str(app.Param2Slider.Value);
        ...
        end
end
```

改变下拉菜单的值后，利用 *VisibleAdjust* 函数调节各自滑条的可见性，并对应改变滑条的值域范围、初始值、标签内容即可。

此处设置的初始值分别为：

脉冲函数：初始坐标 $(128, 128)$

三角函数：初始参数 $f = 0.025, \theta = 0, \phi = 0$

矩形函数：初始坐标 $(128, 128)$、初始角度 $\theta = 0$、初始边长 $128, 128$

高斯函数：初始方差 $\sigma^2 = 100$

## *1.4* 实验结果与分析

改变下拉框选择的函数类型，实验结果分别如下所示：

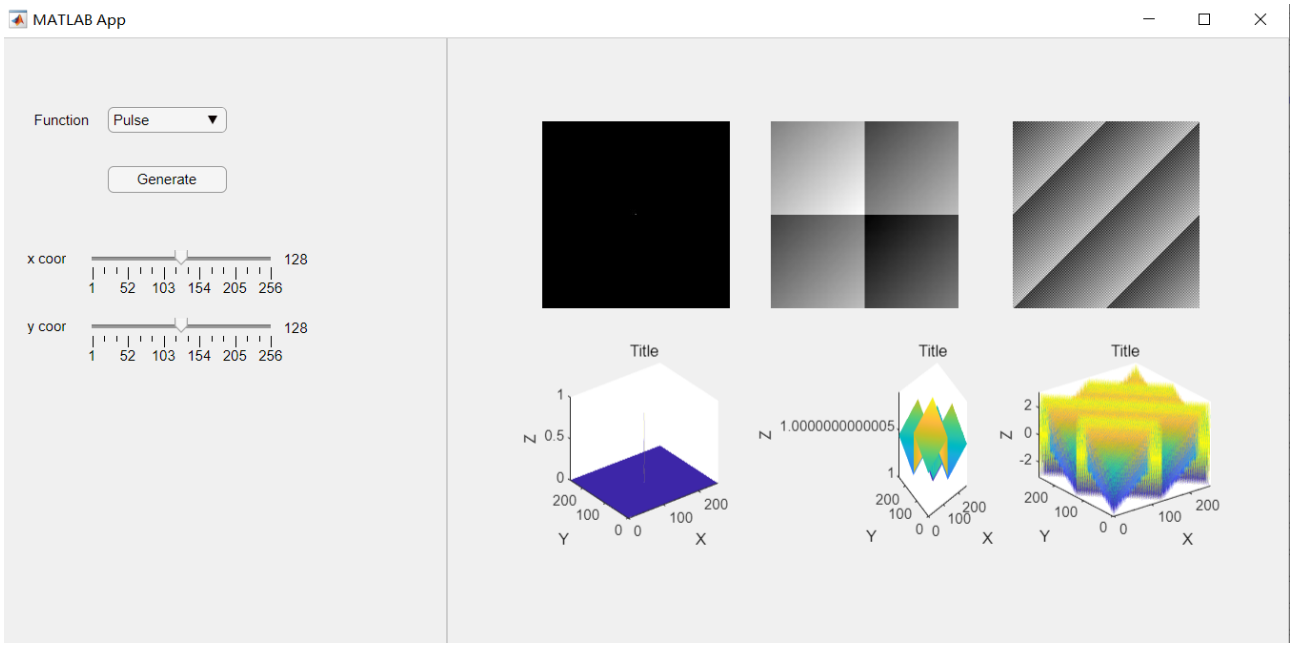


图 2: 脉冲函数生成结果

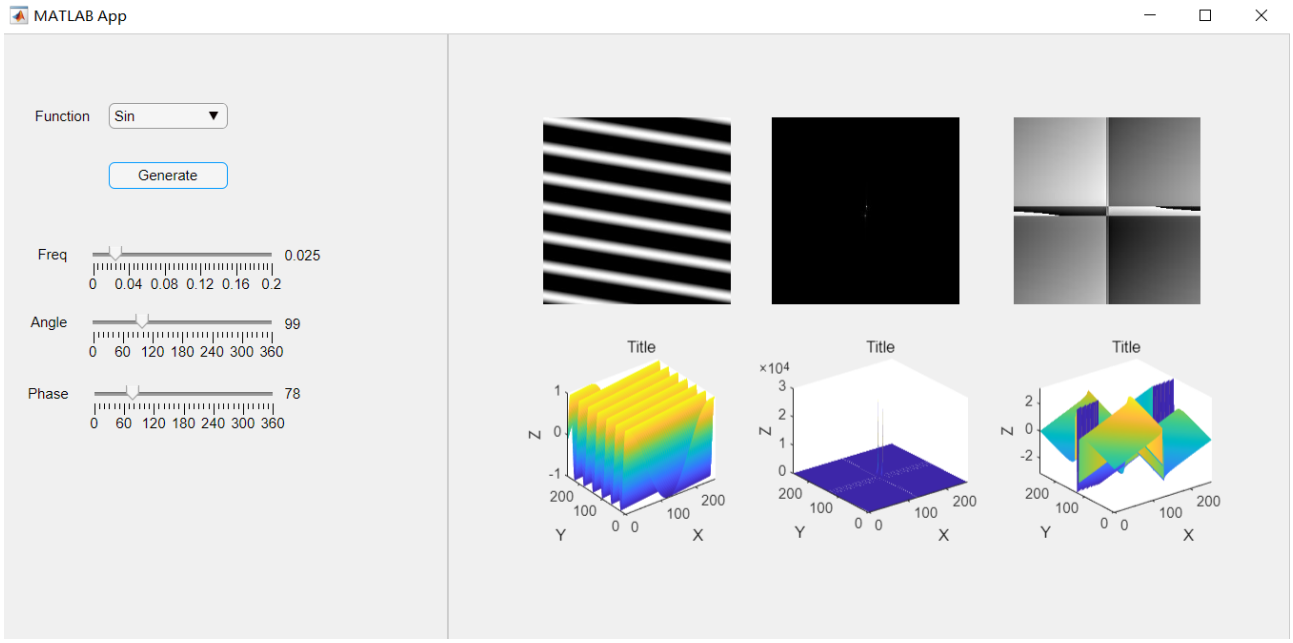


图 3: 二维三角函数生成结果

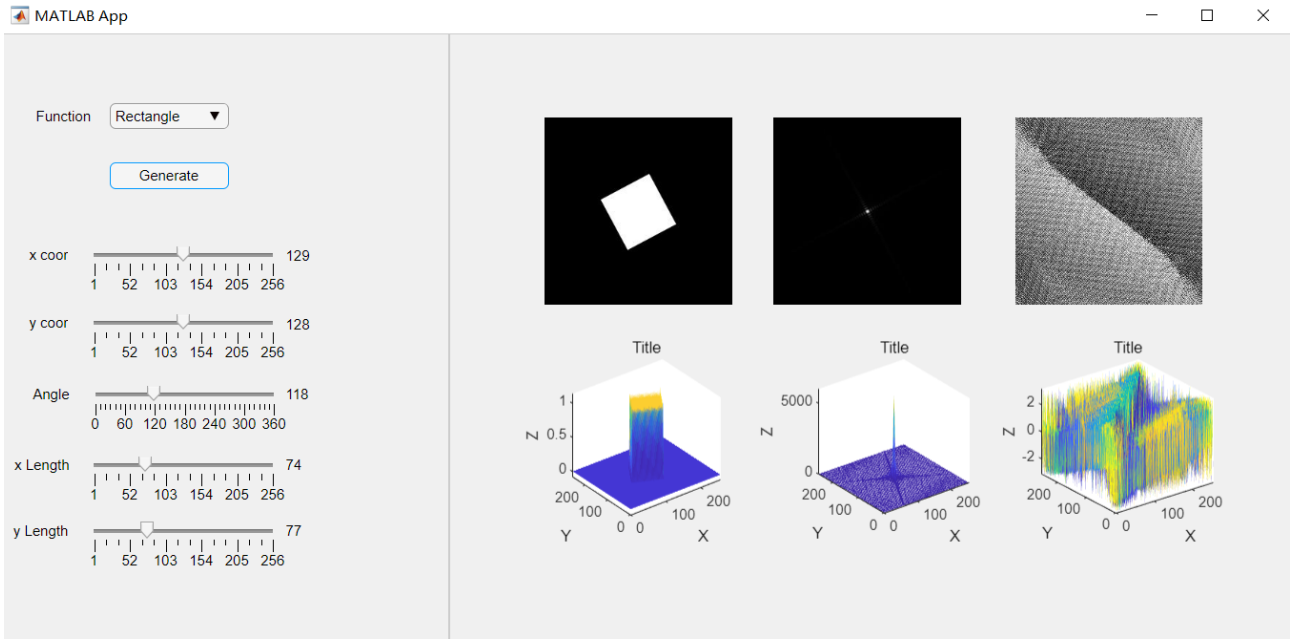


图 4: 矩形函数生成结果

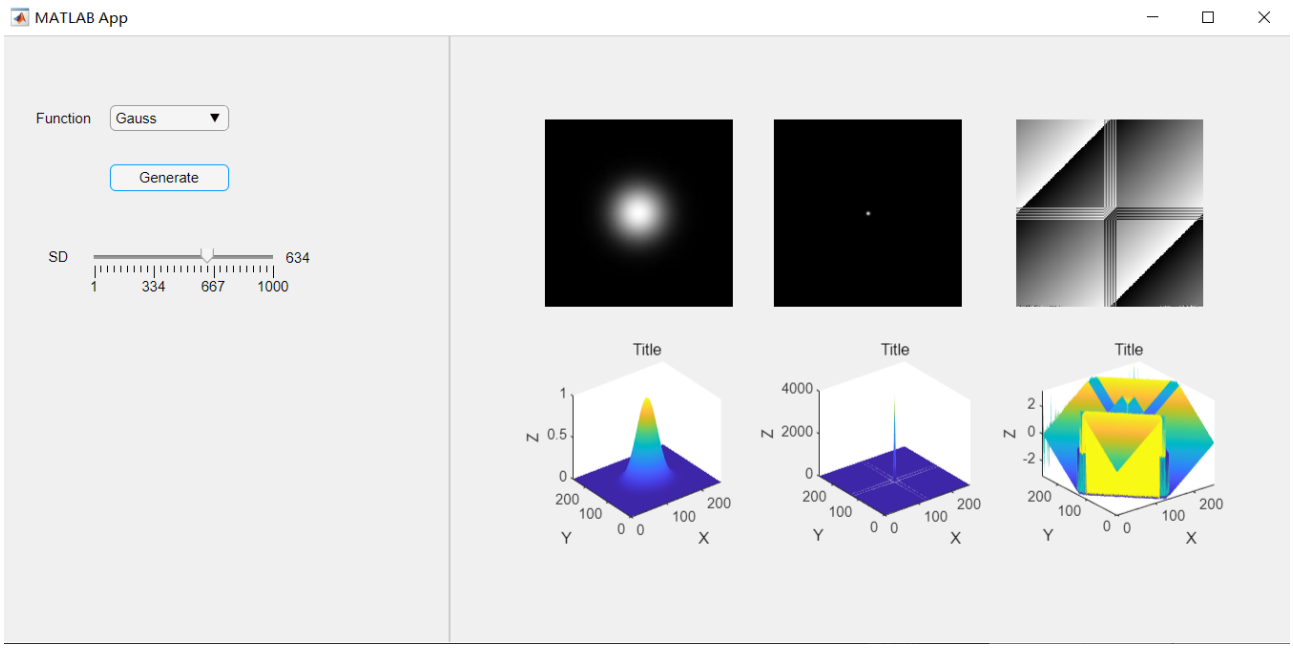


图 5: 高斯函数生成结果

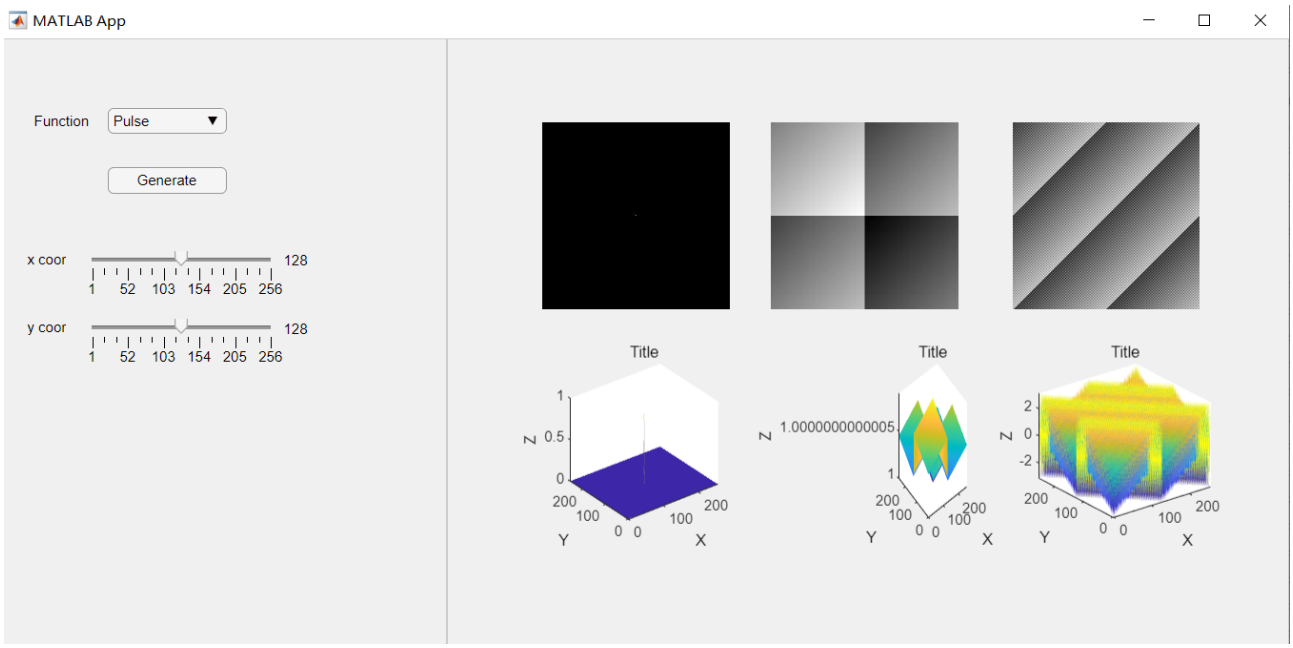


图 6: Gabor 函数生成结果

调节各参数，并与上课提供的 PPT 进行比对可知，生成结果正确，且能够根据参数变化产生正确的结果。

需要注意的是，由于 $surf$ 函数本身的特性，坐标轴精度无法进行调节，故脉冲函数的幅度谱会产生压缩。

# 2 题目二

## 2.1 算法原理与代码分析

根据题目中的描述，该算法可大致分为以下几个步骤：分割图像、最大值提取、对应值计算、图像显示。程序代码如下所示：

```
img = imread('106_3.bmp');
img = im2double(img);
segLength = 16;
DFTLength = 32;
fingerParam = 20;
[height, width] = size(img);
segNum_h = ceil(height / segLength);
segNum_w = ceil(width / segLength);
ROI = zeros(segNum_h, segNum_w);
Angle = zeros(segNum_h, segNum_w);
Period = zeros(segNum_h, segNum_w);

for index_h = 1:segNum_h
    for index_w = 1:segNum_w
        wStart = max(1, (index_w-1) * segLength + 1 - (DFTLength / 4));
        wEnd = min(width, (index_w-1) * segLength - (DFTLength / 4) + DFTLength);
        hStart = max(1, (index_h-1) * segLength + 1 - (DFTLength / 4));
        hEnd = min(height, (index_h-1) * segLength - (DFTLength / 4) + DFTLength);
        segImg = img(hStart:hEnd, wStart:wEnd);
        segImg = adapthisteq(segImg);
        DFTImg = abs(fftshift(fft2(segImg)));
        DFTImg_sort = sort(DFTImg(:));
        if DFTImg_sort(end-1) < fingerParam % No Fingerprint
            continue
        end
        [posX, posY] = find(DFTImg == DFTImg_sort(end - 1));
        if length(posX) ~= 2
            continue
        end
        distance = sqrt((posX(1) - posX(2))^2 + (posY(1) - posY(2))^2);
        Period(index_h, index_w) = 1 / distance;
        if distance == 2
            continue
        end
        ROI(index_h, index_w) = 1;
        angle = atand((posY(1)-posY(2))/(posX(1)-posX(2)));
        Angle(index_h, index_w) = angle;
    end
```

```
39 end
40
41 maxNum = max(max(Period));
42 minNum = min(min(Period));
43 Period = (Period - minNum) / (maxNum - minNum);
44 Period = imresize(Period, 16, 'nearest');
45 imwrite(Period, 'Period.png');
46 figure(1);
47 hold on;
48 imshow(img);
49 DrawDir(1, Angle, 16, 'r', ROI);
```

下进行详细分析。

第 1 行至第 11 行对相应值进行初始化。读取图像至 *img* 变量中，设置分割块大小为 16 像素，DFT 变换块大小为 32 像素，设置判断是否有指纹的阈值为 20。读取图像的 *height*、*width*，计算分割块的数量，并初始化表示指纹状态的矩阵 *ROI*、方向矩阵 *Angle*、周期矩阵 *Period*。

第 13 至第 20 行对图像进行分割。利用等差数列的原理计算分割后图像的四角坐标，由于分割块宽度为 16 像素，DFT 宽度为 32 像素，故在图像四周各取 8 像素进行 DFT 计算，与矩形函数中类似，利用 *min*、*max* 函数控制四角位置确保数组不越界。利用索引切片得分割后图片 *segImg*。

为提高处理效果，利用 *adapthisteq* 函数对图像进行自适应直方图均衡。

第 21 行至第 25 行进行 DFT。调用系统实现的 $fft2$ 函数进行 DFT 变换，并利用 *abs* 函数得幅度值。对计算的幅度结果进行排序，通过比较第二大的值与阈值的关系判断该块是否有指纹。若无指纹，直接跳过该块。经过后续调试，取该参数为 20。

第 26 行至第 29 行进行最大值提取。利用 *find* 函数在原矩阵中索引得第二大的值的位置，并确保共索引到两个值，舍弃其他情况。

第 30 行至 39 行计算对应参数。其中，距离为计算两最大值点之间的欧氏距离，并将其倒数作为周期；方向角度利用反正切函数计算，并赋值至方向矩阵的对应位置。

**需要注意的是，此处进行了一定的特殊处理，舍弃了两峰值点距离为 2 的图片块。这是由于，若不进行该处理，在生成的指纹边缘会产生竖线，这是因为在指纹边缘频率较低，得到的两个峰值点过于接近，使得两点几乎处于同一竖直线或水平线上，产生不正确的结果。此处通过舍弃这些点解决了该问题，在未丢失有效信息的前提下使图像干扰更小，可读性更高。**

第 40 至第 49 行进行图像绘制。对周期图进行归一化后保存，并调用 *DrawDir* 函数绘制指纹方向。此处利用最近邻插值增大周期图的分辨率以便观察。

## *2.2* 实验结果与分析

实验结果如图7所示。可以看到，能够实现较好的方向提取效果。

图 7: 方向提取结果

对应的周期图如图8所示。

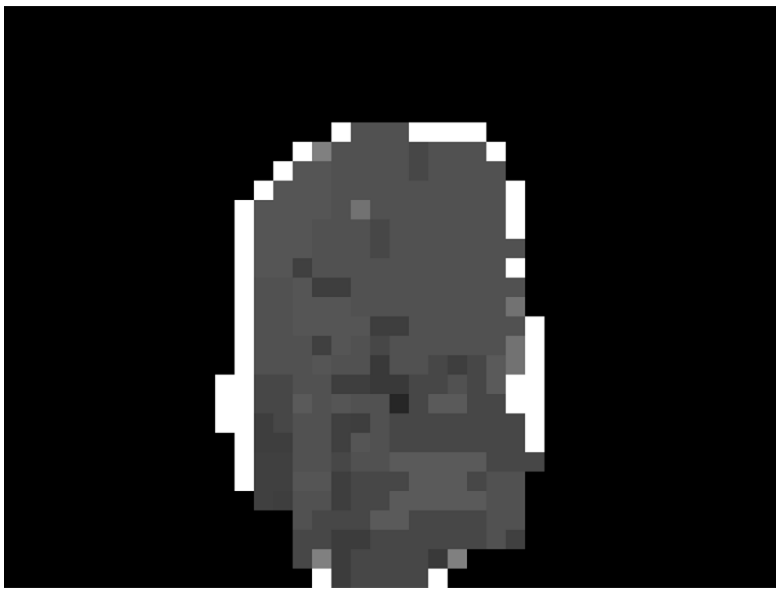

图 8: 周期提取结果

若不采取对边缘的处理方案，所得实验结果如图9所示。明显可以看到，指纹四周产生的大量的错误数据。

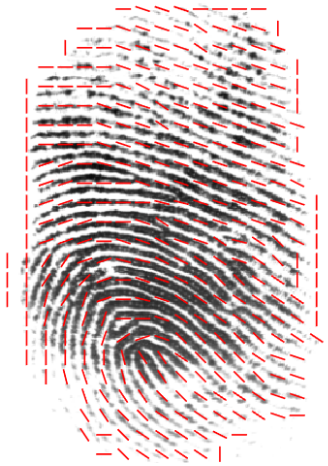

图 9: 方向提取结果 (无边缘消除)

若不采取直方图均衡算法，所得实验结果如图10所示。明显可以看到，产生了更多的错误结果。

图 10: 方向提取结果 (无直方图均衡)

# 3 遇到的困难与解决方案

本次实验原理较为简单，但仍遇到了一些问题，概括而言有以下几点。

- Matlab App Designer 的 surf 函数并不能类似 GUI 直接赋值给图像，而需要利用其定义的 UIAxis 类进行赋值。这一点在网络以及题目中并无相关说明，笔者花费了大量的时间进行探索。此外，该 UIAxis 类也不能控制坐标轴精度。
- 在题目二中，初始生成的方向提取结果较差，边缘处有较多竖线。笔者尝试了利用灰度值、利用位置等方式，效果均不佳。在观察最终的周期图时，笔者发现竖线块与周期图中的白色方块重合度较高，因此笔者尝试利用距离判断去除这些白色块产生的结果，最终取得了很好的效果。

# 4 收获

- App Designer 的进一步使用。
- 指纹方向提取的方法。
- 特殊二维图像空域与频域的相关性质。
- 图像离散傅里叶变换的原理及实现。
- 实验结果的优化与改进。