《数字图象处理》小作业 4

2017011010 杜澍洽 自71

一、 实验任务

1. （题目一）合成 4 种图像，实现 2D DFT 函数用于计算这些图像的 2D DFT，参数通过图形用户界面进行调整。
2. （题目二）寻找 3 幅图片，在每幅图片中分别用四个矩形裁剪出近似的矩形、正弦波、高斯和脉冲图像，施加傅里叶变换并显示效果图。

二、 题目一

计算方法如下图所示，这是上学期信号与系统课程中教授的内容，在函数 my_DFT（）中实现。

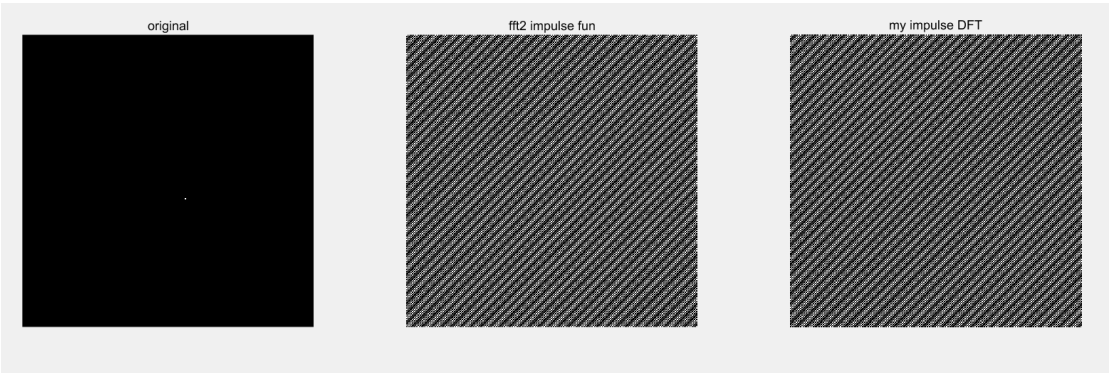
An N-point DFT is expressed as the multiplication $X = Wx$, where x is the original input signal, W is the N-by-N square DFT matrix, and X is the DFT of the signal.

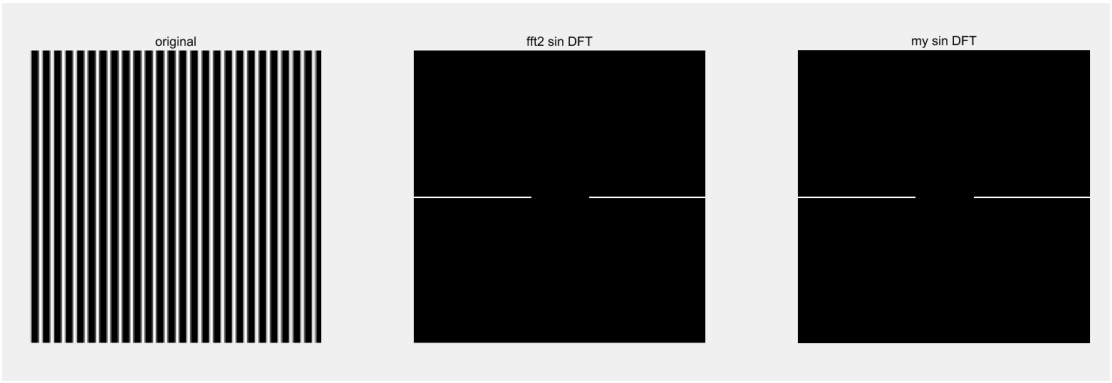
The transformation matrix W can be defined as $W = \left(\frac{\omega^{jk}}{\sqrt{N}}\right)_{j,k=0,\ldots,N-1}$, or equivalently:

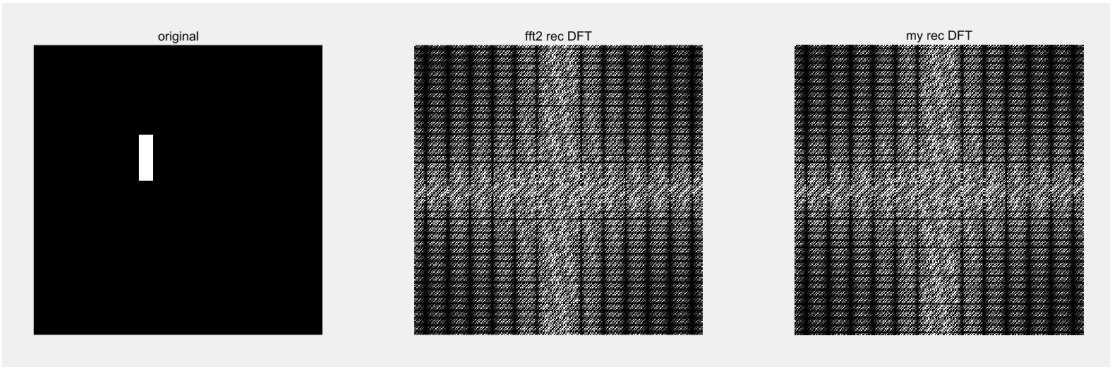
$$W = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix},$$

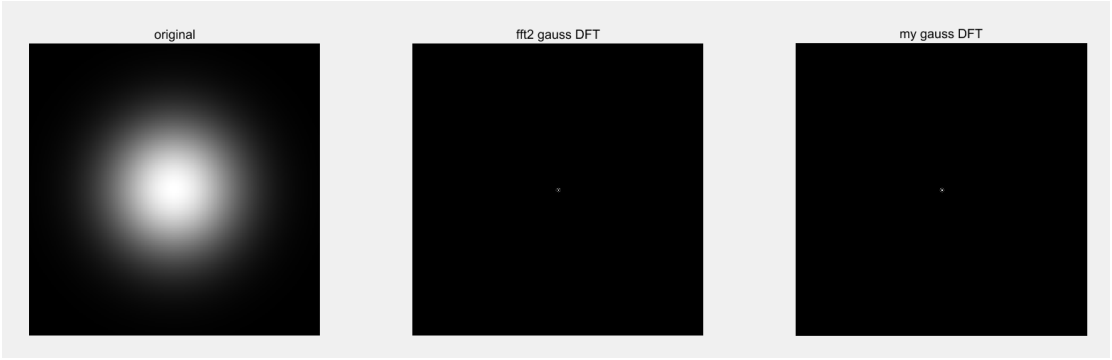
where $\omega = e^{-2\pi i/N}$ is a primitive Nth root of unity in which $i^2 = -1$. We can avoid writing large exponents for ω using the fact that for any exponent x we have the identity $\omega^x = \omega^{x \mod N}$. This is the Vandermonde matrix for the roots of unity, up to the normalization factor. Note that the normalization factor in front of the sum (

生成的四种图像样例及对应的 fft2（）函数和 my_DFT（）函数运行结果对比如下，对比变量表中的数值确定二者完全相同：

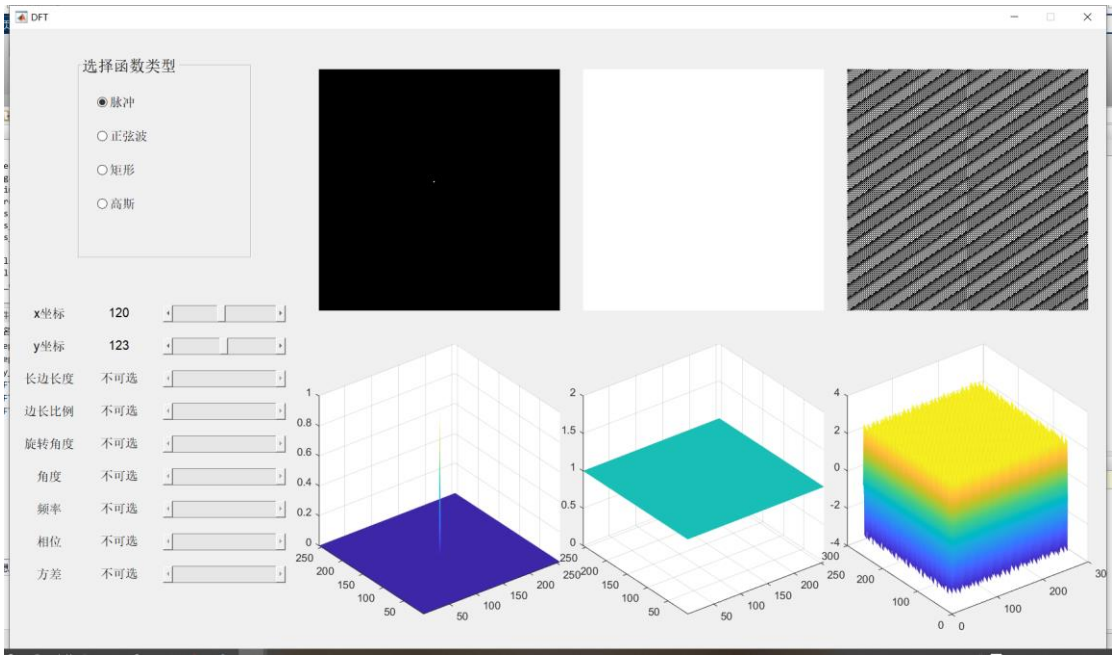


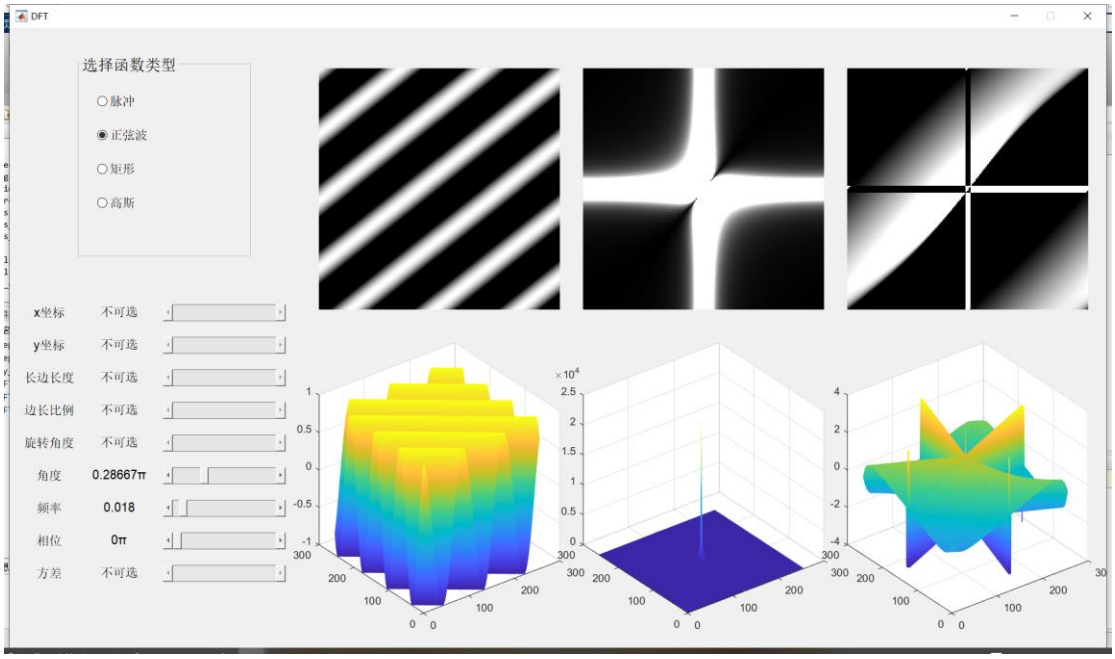

original
fft2 sin DFT
my sin DFT

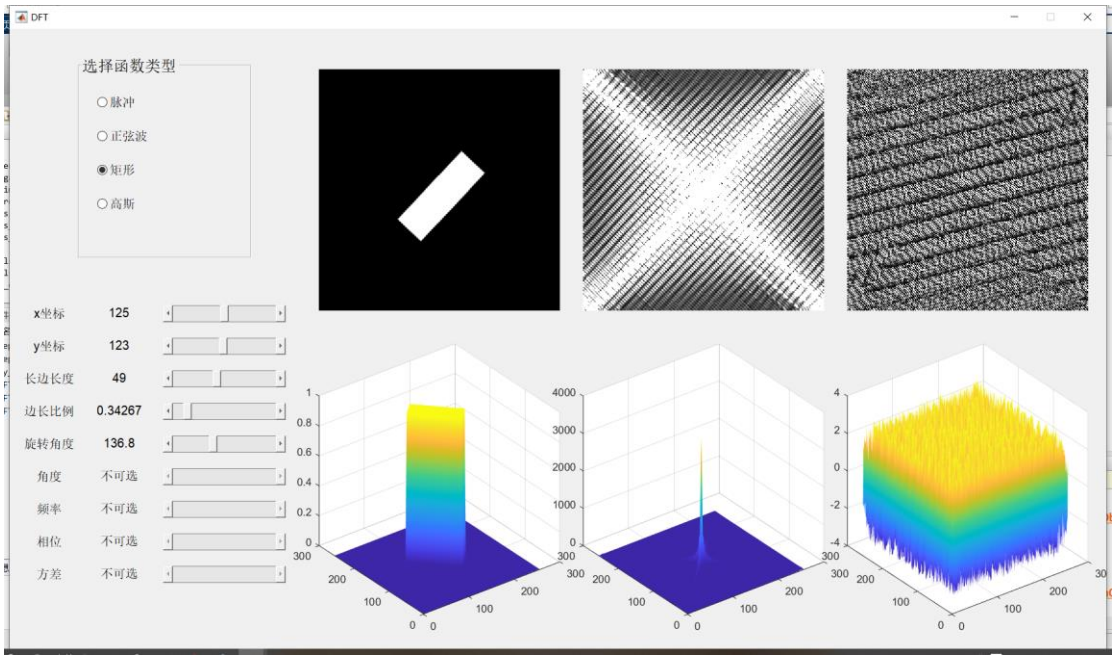

original
fft2 rec DFT
my rec DFT

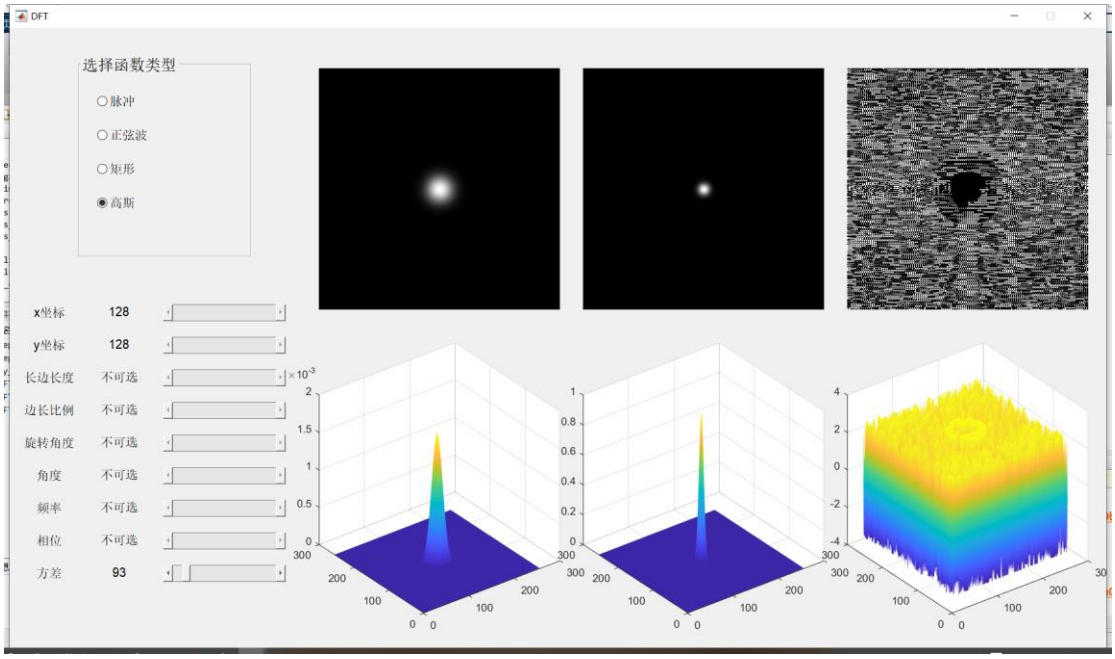

original
fft2 gauss DFT
my gauss DFT

图形界面如下：


DFT
选择函数类型
脉冲
正弦波
矩形
高斯
x坐标 120
y坐标 123
长边长度 不可选
边长比例 不可选
旋转角度 不可选
角度 不可选
频率 不可选
相位 不可选
方差 不可选

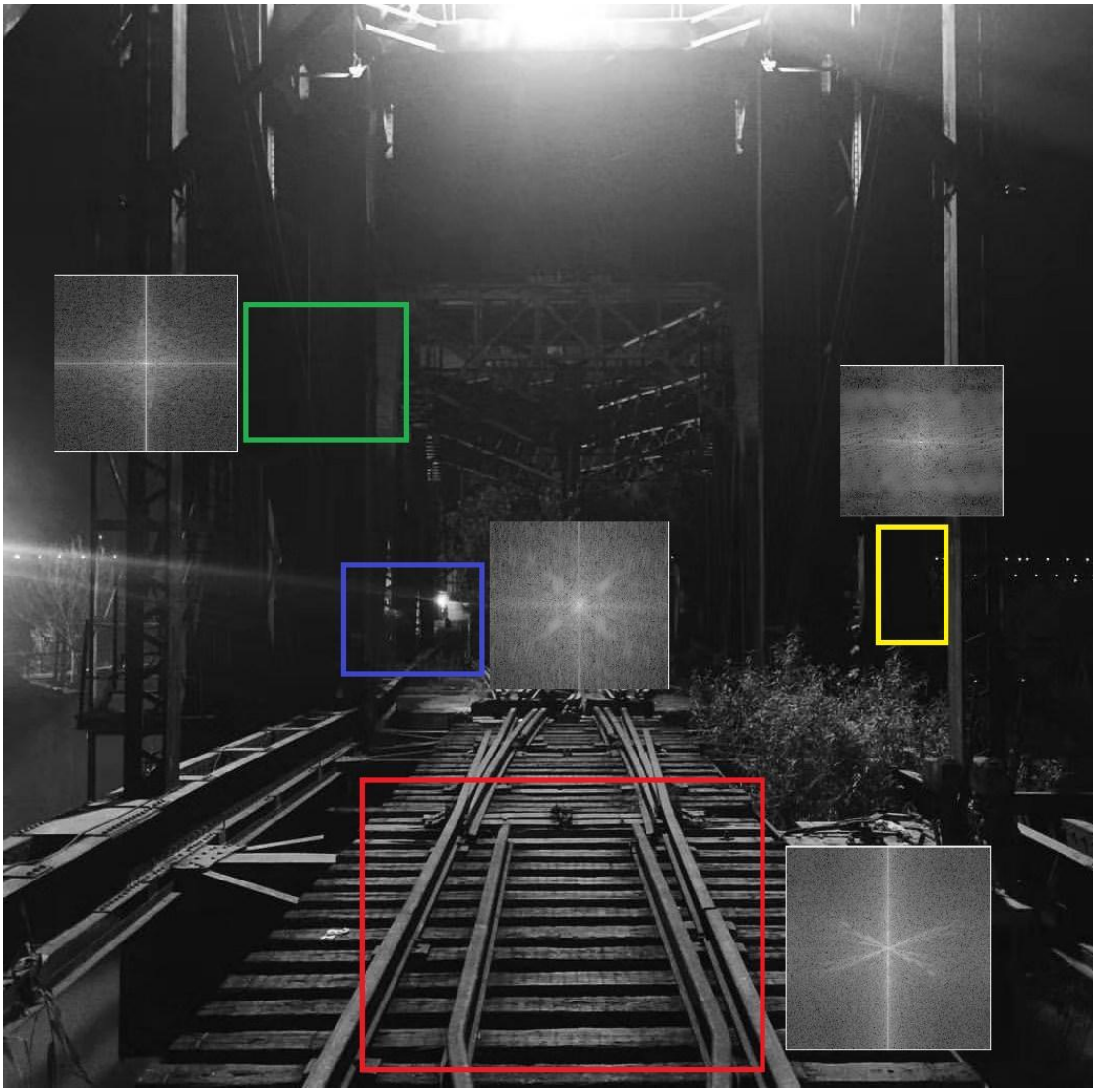

DFT
选择函数类型
脉冲
正弦波
矩形
高斯
x坐标 不可选
y坐标 不可选
长边长度 不可选
边长比例 不可选
旋转角度 不可选
角度 0.28667π
频率 0.018
相位 0π
方差 不可选

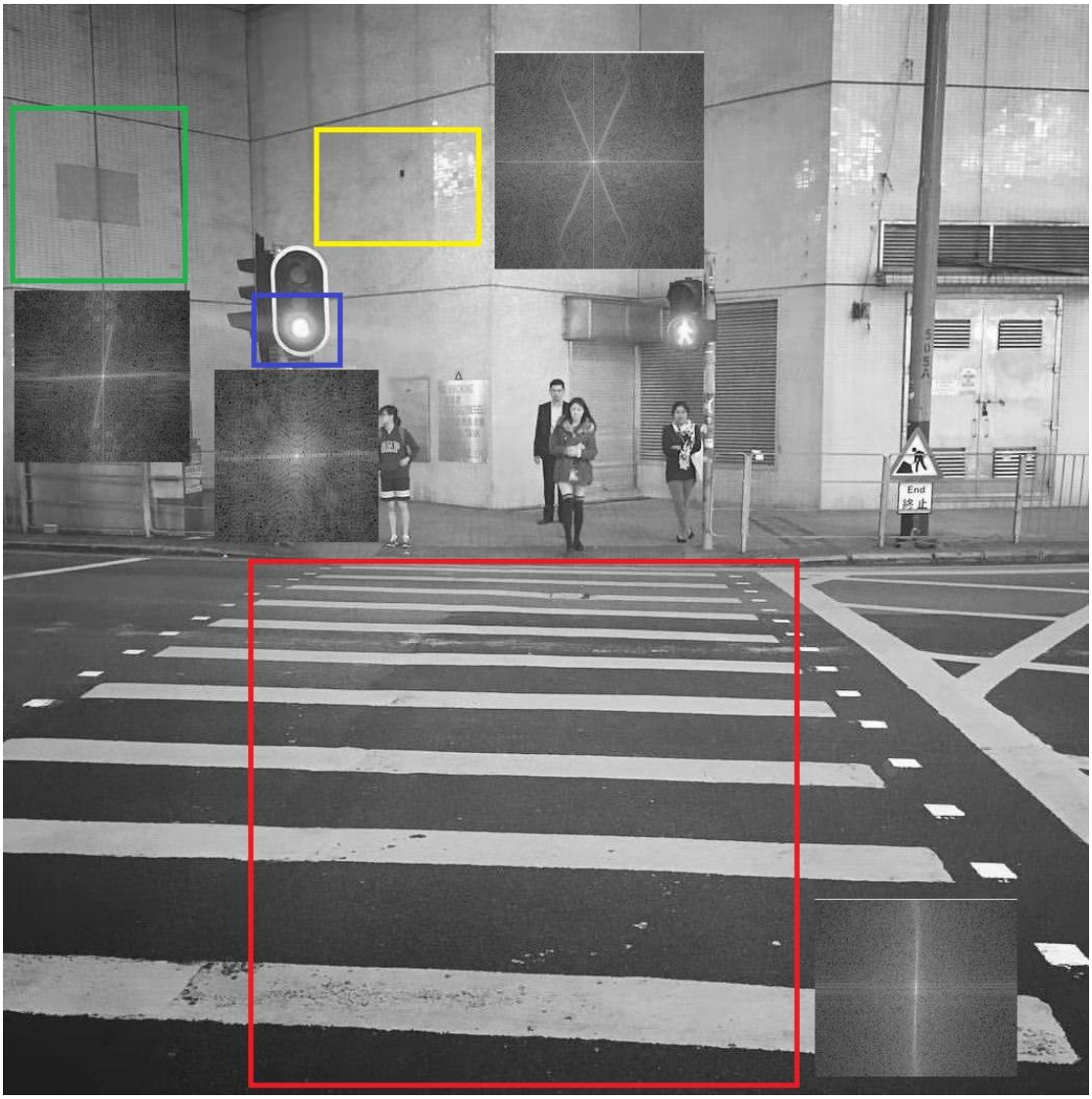




三、 题目二

标注后的三幅图像如下：




End
終止

将现实图像与理想图像对比，可以看到二者大致类似，但现实图片中的干扰显然更多；而且由于现实图像形状不够规则，灰度分布也不能和函数表达式完全吻合，变换后的结果也不及理想情况规则。

四、 实验总结

关于自己的 2D DFT 算法，我一开始写了一个运行时间 7 秒左右的算法，在实现图形界面的时候就意识到这个速度不能完成拖动进度条立马显示新图像的需求，然后才开始进一步优化速度，最终想到了信号与系统课程上的内容，解决了这个问题。个人觉得寻找第二题中合适的图片是很有意思的过程。