# MATLAB 图像处理实验报告

吴同　　无 93　　2019013217

（题目为 章_节_题号 格式）[原创性声明附在最后]

1_3_1.

查看相关函数，内容略。

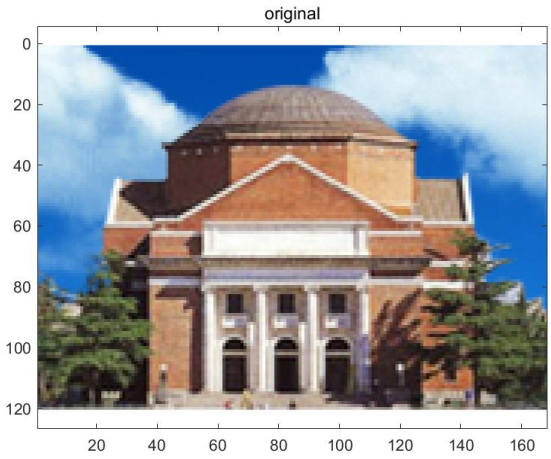
1_3_2(a).

具体代码详见附件 PP1_3_2_a.m.

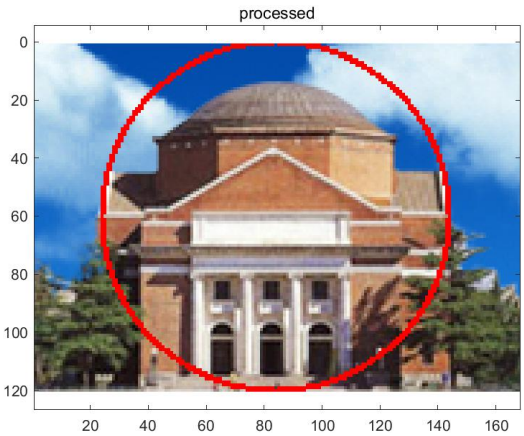
解决方法：首先对图片取中心，获得圆心；然后取图片的高、宽之中较小的值作为直径。再者遍历每个像素点，对于到中心距离和半径差的绝对值小于 1 的点进行变红处理。

结果如下：

原图




处理后的图



最终显示时使用了 axis equal 使画出来的圆不扭曲。

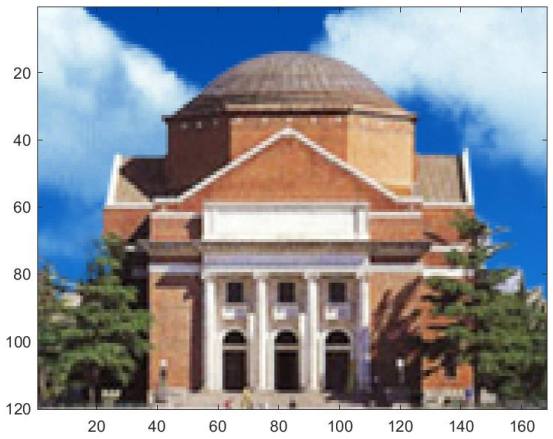
1_3_2(b).

具体代码详见附件 PP1_3_2_b.m.

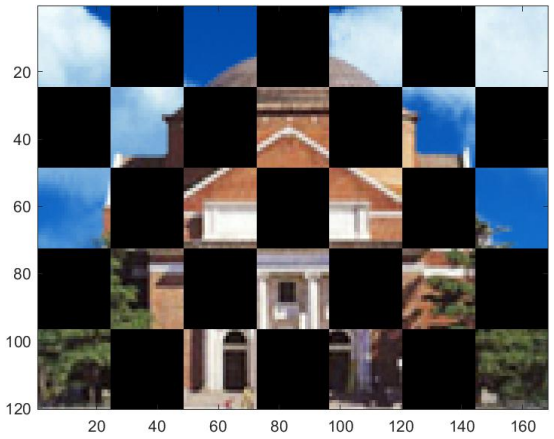
解决方法：首先获取图片的高、宽最大公约数作为黑白格子的宽度；接着利用这个宽度分别求出沿着高方向和宽方向的黑白格子数量；然后构造黑白格子规模的 1 矩阵，黑白格子数量规模的 0，1 间隔矩阵，利用 b 和 a 的张量积就可以获得单位黑白格子；最后用原图片乘（数组乘）单位黑白格子就可以得到目标。

结果如下：

原图




处理后的图




总结：这题解决很关键的一点就是认识到每块黑白格子本质是一致的，从而很容易想到运用 kronecker 积的办法和 matalb 强大的矩阵运算能力快速完成图像处理。

2_4_1.

图像预处理中减去 128 的步骤可以在变化域中进行。首先，从直觉上看，在原图中减去

128 本质上是减去了一个直流分量，在变化域中左上角的的元素本身就反映了直流分量，相差的可能只是一个系数问题。从 DCT 变化的公式来看，

$$C' = D(P - 128)D^T = C - 128 * DOnesD^T,$$

其中$C'$为减去 128 的系数，C 为没有减去 128 的系数，Ones 为全 1 矩阵

$DOnesD^T$是一个确定值，为左上角元素为 8，其余元素为 0 的矩阵。因此在变化域中只需要将系数矩阵左上角元素减去 128*8 就可以由相同的效果。

具体代码详见附件 PP2_4_1.m.

解决方法：利用图像不减去 128 而在变换域进行上述处理的系数矩阵逆变换的结果和原图像减去 128 的结果进行比较。

结果如下：

```
error =

   1.0e-12 *

  列 1 至 7

   -0.0711   -0.1137   -0.0284   -0.0568   -0.0568   -0.0568   -0.0853
   -0.0568   -0.0995   -0.0142   -0.0426   -0.0426   -0.0426   -0.0711
```

可见在误差范围内是一致的。

2_4_2.

具体代码详见附件 PP2_4_2.m.

结果如下：

```
error =

   1.0e-12 *

  列 1 至 7

   -0.2274    0.1528    0.1145   -0.1243   -0.0496   -0.4157    0.3162
   -0.2842   -0.0342    0.0769   -0.0275   -0.0218   -0.1545    0.0249
```

可见在误差范围内是一致的。

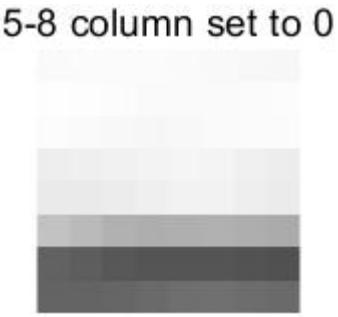
2_4_3.

具体代码详见 PP2_4_3.m.

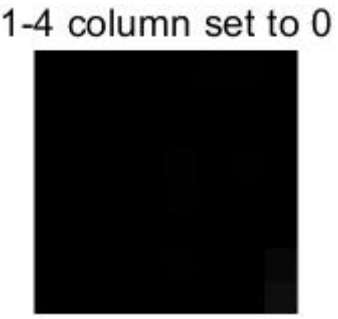
结果如下：

原图


后四列置为 0




前四列置为 0



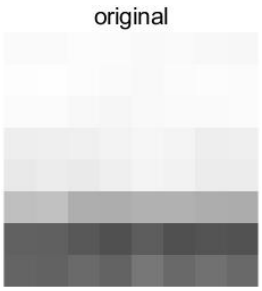

理解和总结：系数矩阵靠左、靠上的元素表示的是直流和低频分量，在图片中往往值比较大；而靠右靠下的代表高频分量，在图片中的值往往比较小。如果将 1-4 列的值设为 0，那么直流和低频信号都没有了，仅剩下值很小的高频分量，因此几乎成了“黑图”；而如果将 5-8 列的值设为 0，仅去除了高频分量，对原图没有很大影响；但是也可以看出相比原图处理后的图在颜色上更加平滑。

2_4_4.

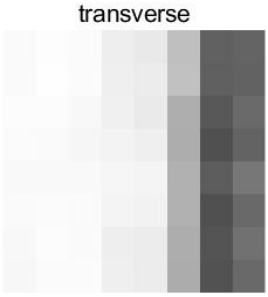
具体代码详见 PP2_4_4.m.

结果如下：

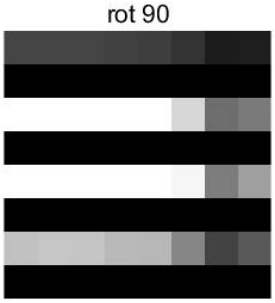
原图



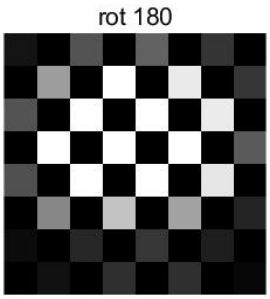

转置




旋转 90 度




旋转 180 度




理解和总结：将系数矩阵转置在原图表现得结果是将原图进行转置，这可以通过 DCT 公式看出。DCT：

$$C = DPD^T$$

$$\therefore C^T = DP^T D^T$$

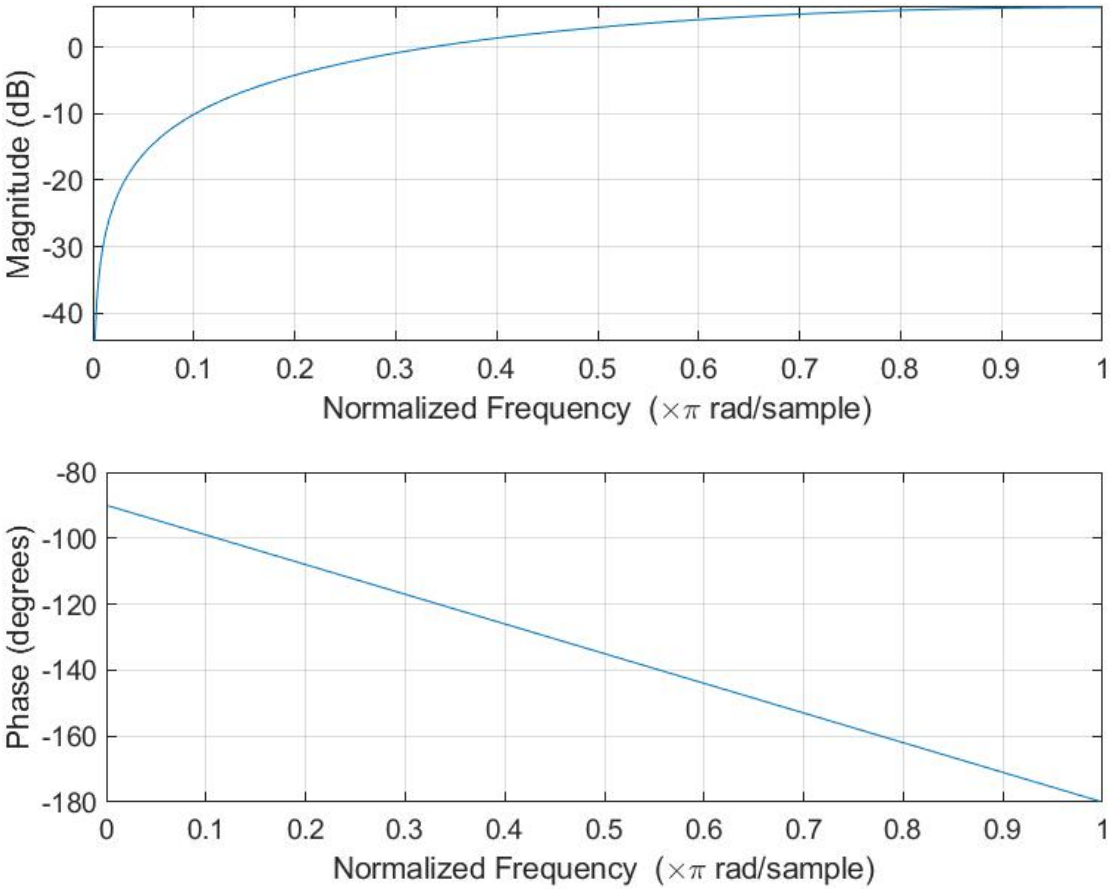
将系数矩阵旋转 90° 后，直流分量旋转到左下角，使行直流高频分量变得很大；原来

右上角较小的值旋转到直流分量，使得直流分量变得很小。因此会呈现上面的图像。将系数矩阵旋转 180 度类似：值最大的直流分量旋转到了右下角，使得二维高频分量很大，而右下角很小的值旋转到左上角，使直流分量变得很小，因此呈现出如上结果。

2_4_5.

具体代码详见 PP2_4_5.m.

频率响应如下：




可见该系统是一个高通滤波器。DC 系数先进行差分编码再进行熵编码，说明 DC 系数的高频分量多。

2_4_6.

DC 预测误差的二进制位数就是 Category 值。（其中 0 为例外，对应 Category 值为 0，负数用 1 补码）因此可以得到计算公式

$$\text{Category} = \log2(\text{abs}(\text{error})) + 1, \text{error!} = 0;$$

$$\text{Category} = 0, \text{error} = 0.$$

2_4_7.

我所知道的 zig-zag 扫描方法有：

A. 循环直接法。直接写一个按照 Zig-zag 遍历思路的循环。这个方法写起来在判断边界上稍显复杂，没有利用到 matlab 的特性。

B. 访问表法。上网查找 8*8 的 zig-zag 访问表，利用 matlab 下标访问特性，可以很方便地实现 zig-zag 扫描。这个方法的缺点是需要使用 zig-zag 访问表，而且实现对象是固定大小的，不同大小的矩阵需要使用不同的访问表。但是对于这里的问题使用该表没有任何问题，在代码实现上也非常简单。

我选择的方法是访问表法，访问表来源于网络
https://blog.csdn.net/weixin_33757609/article/details/91974234.

具体代码详见附件 zigzag.m.

2_4_8.

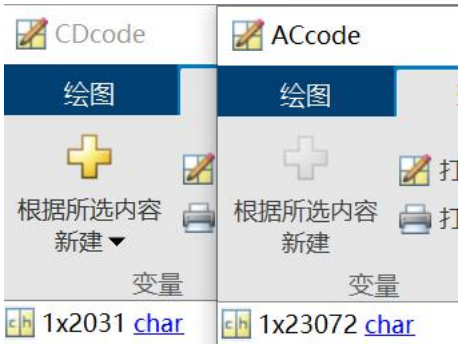
按照图像分块、DCT、量化、zig-zag 扫描的顺序进行代码编写。具体代码详见附件 photoProcess.m.

2_4_9.

具体代码详见附件 coding.m.

2_4_10.

输出码流大小如下：




输入文件大小如下：

可见输出码流为 2031+23072 = 25103；输入文件大小为 120*168*8=161280。因此压缩比为 6.4。

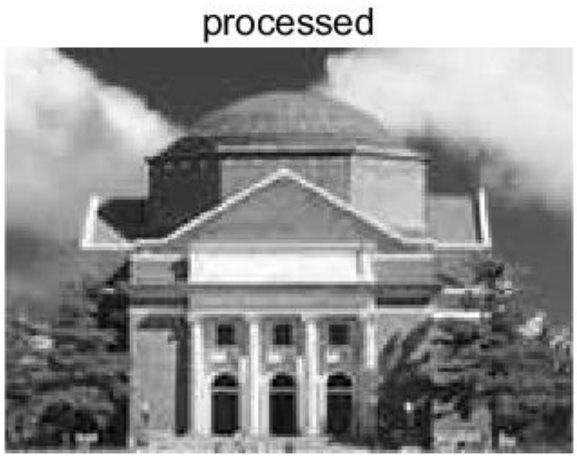
2_4_11.

JPEG 解码具体代码详见附件 decoding.m 和 PP2_4_11.m.

结果如下：

原图


编码解码处理过的图



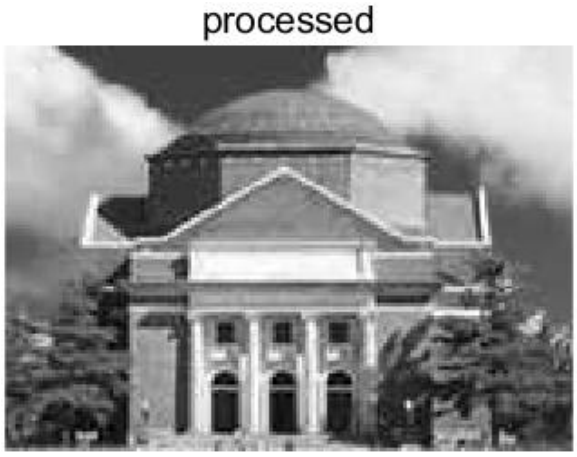

从视觉效果上可以看出编码解码效果是比较好的，没有很大的差别，有一些的模糊。
从 PSNR 值看，两图的 PSNR = 34.1854，可以看出还是比较好的。

2_4_12.

具体代码详见附件 PP2_4_12.m.

结果如下：

原量化步长的图




减小量化步长的图

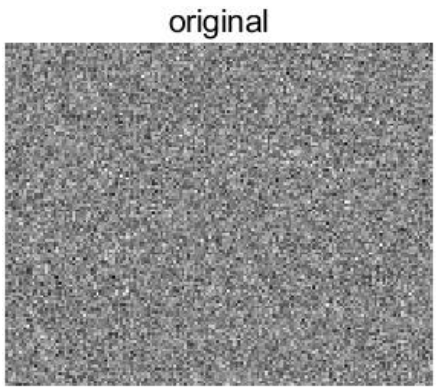

单从视觉上看，减小量化步长的图比之前的图更加清晰。从 PSNR 评价角度上看，减小量化步长的图的 PSNR 值为 39.1991，比之前的要大，因此要更好。

2_4_13.

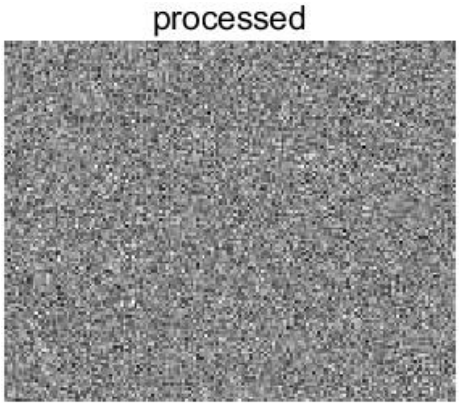
具体代码详见 PP2_4_13.m.

结果如下：

原图

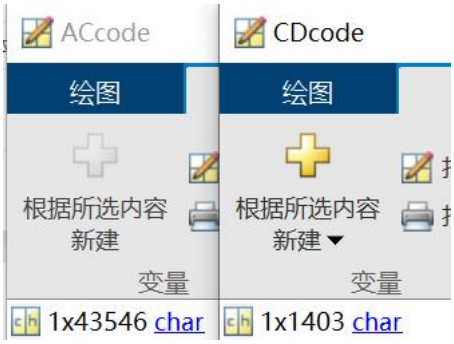


编解码处理后的图




从视觉效果上，似乎看不出差别（因为原图本身就是不规律的）。从 PSNR 值角度，此图的 PSNR 值为 25.9577，比之前的要小，说明在 PSNR 评判标准上它是效果更差的。这是因为雪花噪声图像的高频分量是很大的，而在量化过程中高频分量的量化步长很大，因此会造成很大的量化误差，导致 PSNR 评价结果很差。

另外，输出码流大小如下：




输入图像大小如下：


因此压缩比为 128*160*8/(43546+1403) = 3.645.可以看出也没有之前的好，这也因为雪花噪声含有很多高频分量，导致量化后还有很多系数非零，导致 ACcode 大量增加，从而使得输出码流大大增加。

3_4_1.

具体代码详见 PP3_4_1.m,SIH.m,iSIH.m.

结果如下：

利用空域隐藏方法和提取方法获取编解码图片中的信息会会出现很多错误，以下为部分（偏差）：

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

说明空域隐藏和提取具有扛 JPEG 编码的能力。

这是因为空余隐藏对图片本身的精度有很高要求，而 JPEG 会有产生一定程度的损失，因此很容易造成信息的丢失和错误。

3_4_2.

替换每个 dct 系数的最低位的具体代码详见 photoProcessWithH1.m,decodingWithH1.m.

替换部分 dct 系数的最低位的具体代码详见 photoProcessWithH2.m,decodingWithH2.m.

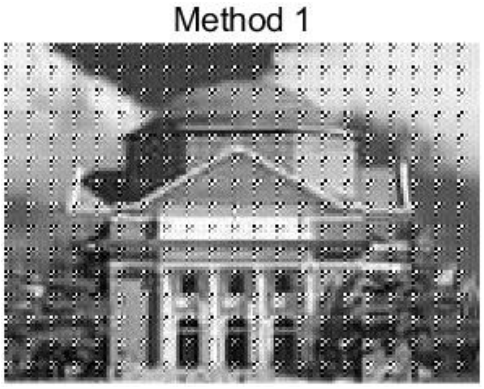
将 1，-1 的信息序列写入 zig-zag 块最后一位后的具体代码详见 photoProcessWithH3.m,decodingWithH3.m.

调用具体代码详见 PP3_4_2.m.

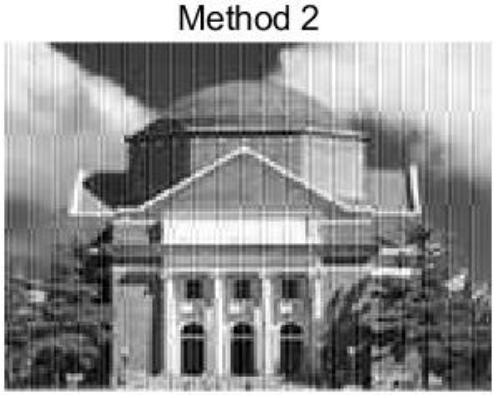
以上三种方法中的第一种和第三种是确定的，而第二种是由我们自己设计的。我考虑的是每个 dct 系数矩阵选取第一行进行信息隐藏，其它行均不进行信息隐藏。

隐藏信息后的图像如下：

方法 1




方法 2



方法 3


由上面的视觉效果可知方法 1 和方法 2 都可以明显看出和原图不同，隐蔽性不太好。而方法三的隐蔽性较好。这是因为方法 1 和方法 2 都是强制对量化后的系数矩阵进行修改，很多本该是零的高频系数都被置为 1，而其量化步长又很大，因此会多出很多高频分量。而方法 3 是在最后一个非零处后加入信息，这样一方面可以减少很多高频信号，另一方面由于加入的位置在非零之后，给图片带来的影响不会非常大（因为已经有差不多频率的分量了）。从 PSNR 评价角度看，方法 1，2，3 的 PSNR 值分别为 18.5435，28.3135，31.9142，和分析一致。（方法 2 比方法 1 好是因为只改变了部分系数值，受此启发可以按照 zig-zag 扫描顺序或者 QTAB 元素大小顺序隐藏信息，从而获得更小的误差）

此外，方法 1 的压缩比为 120*168*8/（66294+2032）= 2.36，方法 2 的压缩比为 120*168*8/（2032+28006）= 5.36，方法 3 的压缩比为 120*168*8/（2031+24017）= 6.19.大小顺序和视觉效果是一致的。这是因为多出来的高频分量会带来大量的 ACcode。这也印证了上面的分析。

4_3_1.

具体代码详见附件 train.m,PP4_3_1.m.

样本人脸大小不一时，不需要将图像调整为大小相同，因为提取的特征是颜色的类型分布，并且提取过程中有归一化处理，不同大小的图片都能提取出相同大小的特征向量，所以不需要对原图像进行大小调整。

当 L 取值不同时（3、4、5），对于三个颜色通道的量化是不同的，例如 L=3 对于每个通道量化为 8 个颜色，L=4、5 分别为 16、32 个颜色，因此 L 较大的量化无非就是将 L 较小的量化进行进一步的划分而已。因此，对于 L=3、4、5 的特征向量，若将其 reshape 成 3 维向量，那么同一张照片的累积分布是相同的，例如 L=4 的（1：2，1：2，1：2）概率会和 L=3 的（1，1，1）概率相同。

4_3_2.

具体代码详见附件 PP4_3_2.m,faceDetection.m.

解决方法：

我对人脸检测分了 4 个步骤：首先判断整张图片是否是人脸；如果不是进行下一个步骤——划分，将图片取出小的一块进行判断。这个会迭代进行，直到达到给定的深度；接着是对有较高重叠度的判定部分进行筛选，选出符合度最高的集合；最后是对目标区域进行微调。

第一步的想法源于训练集——全都是整张人脸图，保证这种情况在考虑范围内。

第二步最开始我采用的是统一的划分方式：划分的子块高宽均为母块的 1/2，高、宽方向的划分块数均为 5，移动步长由以上两者决定（和卷积核中的概念相似）。但是结果表明第一步的划分还是太大了，很容易出现一个框里由多张脸的情况，调整距离阈值也不能达到很好效果。因此我修改了第一步的划分方式，改成子块高、宽均为母块的 1/10，而步长为子块的 1/2，实现更小的范围的检测。接着对大于距离阈值的块进行进一步的划分，（此时的划分方式是最开始的方法）以便找到可能存在的人脸。划分部分有一个输入的深度参数，如果深度参数达到最大值，则停止进一步划分，认为这里不存在人脸。未达到深度最大值的块都会进行检测，如果距离小于阈值，那么认为这里有人脸，将其记录；否则认为可能存在阈值进行进一步划分。

第三步是由我第二步的方法决定的。由于在第二步中移动步长可能很小，导致被认为含有人脸两个子块有很大的重叠部分，如果仅仅只是对所有被认为含有人脸的子块画边框处理，那么会出现很多重叠的矩形。第三步就是对这些重叠的矩形进行筛选，避免高度重叠的出现。采用的方法是对高度重叠（用两个矩形中心距离判定）的矩形进行选择，选出更小距离的。

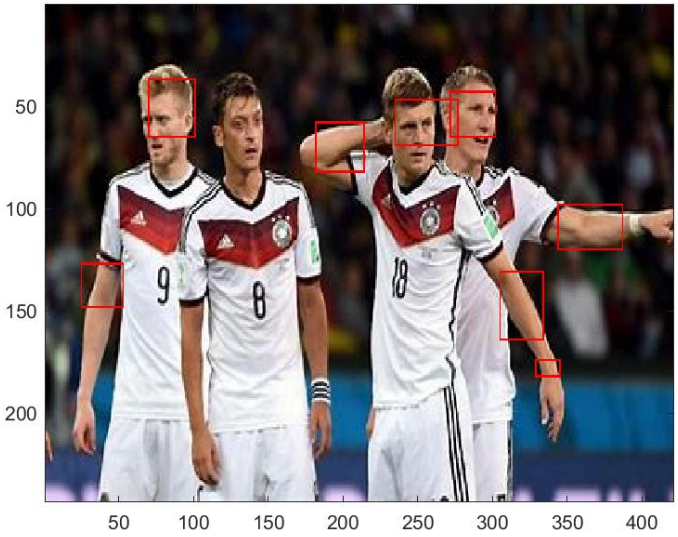
第四步其实可有可无，主要是对前三步的结果进行一定的优化，使矩形更好地框住脸。采用的方法是多次对矩形框图的左上角横坐标、纵坐标、高、宽进行高斯随机采样（期望分别是前三步得到的左上角横坐标、纵坐标、高、宽，标准差分别为高的 1/6，宽的 1/6，高的 1/3，宽的 1/3），选出距离最小的。

结果如下：

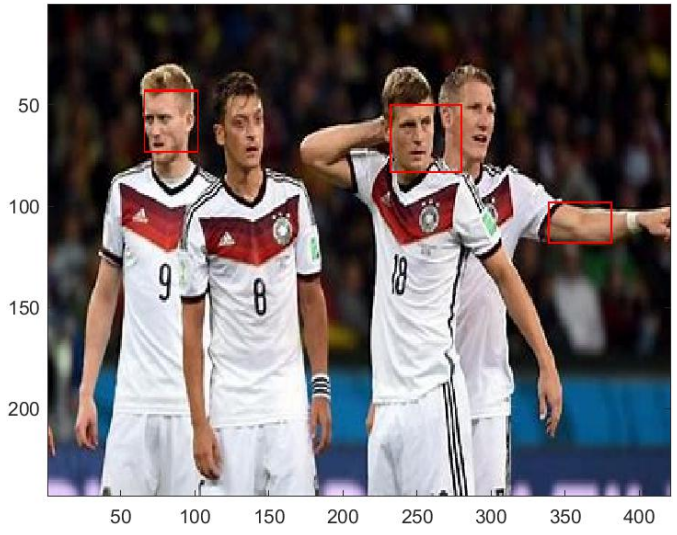
原图


L=3


L=4




L=5




对比 L=3、4、5 时的检测结果，可见随着 L 的增大，检测到的“人脸”越来越少了：错误检测少了，未检测到的增多了。这是因为随着 L 的增大，特征向量的维度增大了，颜色种

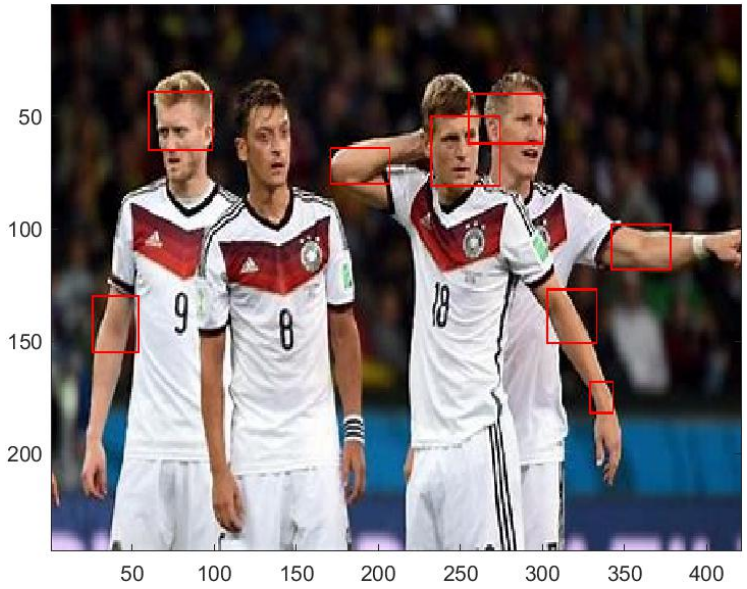
类变多了，也即对颜色的要求更加精细了。原来错误检测的由此减少。而由于训练样本的单一、数量少，人脸标准向量本身就过拟合了，导致未检测到的增多了。

4_3_3.

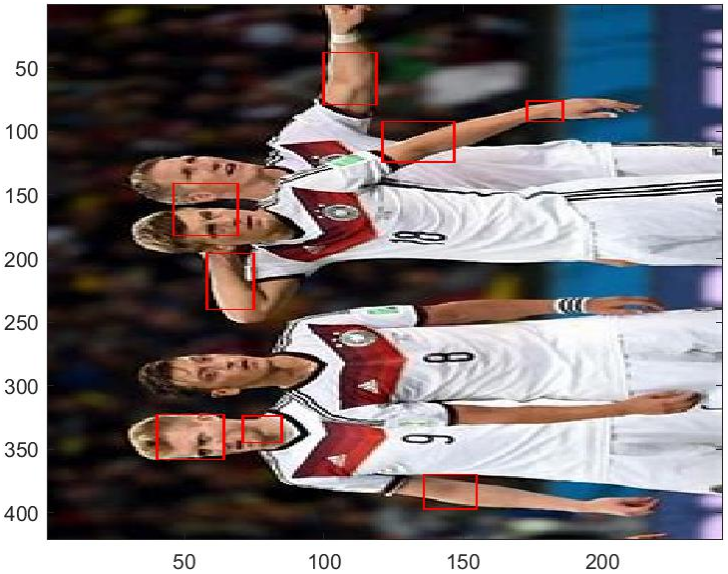
具体代码详见附件 PP4_3_3.m.

结果如下：

原图

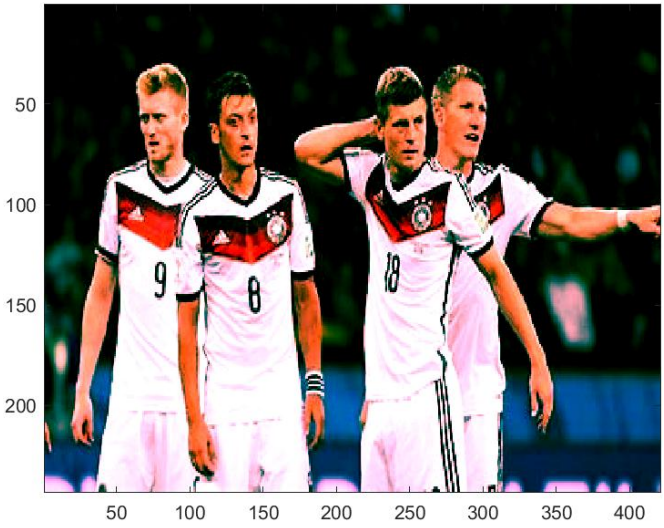



旋转




拉伸


颜色调整




由上面结果可知旋转、拉伸并不会对结果产生较大影响，而改变颜色会严重影响结果。这是因为此人脸检测方法是基于颜色特征向量的，旋转、拉伸都没有对颜色产生影响（虽然可能对分块的检测产生影响），因此最终的结果不会有较大变化。而颜色调整直接改变了其特征向量，因此会严重影响检测结果。

4_3_4.

我用训练集的照片核人脸标准进行对比，发现距离很大的有皮肤很白或光线很强的以及皮肤很黑或者光线很暗的人脸，光线强度合适且人脸皮肤偏黄、棕的人脸距离很小。这个训练集对光线强度合适且人脸皮肤偏黄、棕的人脸明显有过拟合的趋势，而产生过拟合的原因是光线强度合适且人脸皮肤偏黄、棕的人脸很多，皮肤很白或光线很强的以及皮肤很黑或者光线很暗的人脸很少，因此如果重新选择训练标准，应该取每个人脸的特征向量的加权平均。

原创性声明：

图像处理实验中，除了 zig-zag 扫描的 8*8 访问表是复制网上内容外，其余均由自己完成，没有参考网上资料，没有与同学交流。