计算几何实验报告

王弘宇、肖阳、曹云

[问题背景]

目前在图像的信息隐藏领域大致分为空域和频域两类方法，其中空域的方法不能抵抗应用普遍的 jpeg 压缩或者简单的集合变换，而频域的大部分方法对于几何变换的抵抗效果也相当不好，一些隐藏效率较高的方法甚至对简单的频域或空域变换都不能保持不变。而参考文献中的 Constellation 法则是一个相对鲁棒性较高的方法，原论文中没有给出具体的实现方法，我们的实验主要是依据该论文思想进行了一定程度的发挥和创新。

[算法及原理]

主要利用的是频域变换的系数在各种小幅度几何变换内保持不变的性质和频域峰值、点集的凸性对于几何变换保持不变的性质。几何变换主要是指各种仿射变换。

隐藏算法过程为：

1. 对原图进行二维离散余弦变换。
2. 在原图等大的空间内撒播正态分布随机数，每个随机数占据一定的空间，直到连续若干步内找不到空间可以放置随机数。
3. 对该随机点集逐层求凸包，按照每层凸包的单调多边形链在原图上设置极值表示待隐藏信息，每一层都从最左上点逆时针开始。其中某点极值的求法为周围八点的平均值加上 4 倍的标准差。在 R 层的极值表示 1，在 G 层的极值表示 0。
4. 对隐藏后的频域图进行二维反余弦变换。

检测算法过程为：

1. 对待检测图进行二维离散余弦变换。
2. 对每一点进行判断，是否属于极值点。
3. 解码。过程基本同隐藏算法第三步。

[系统设计]

采用 MDI 多文档结构，设计了 CDib 类对 24bit 位图进行操作，和 MATLAB 的接口使用 mwArray 数据类型。Dib[0]储存原图，Dib[1]储存 DCT 变换后的频域图形，Dib[2]储存随机点集的位置，Dib[3]储存隐藏后的频域极点示意图，Dib[4]储存反变换后的已隐藏了数据的图像，Dib[5]储存从待检测图像中所恢复出的极点和其代表的二进制位。

[实现过程中遇到问题及解决对策]

1. DCT 变换后低频分量集中在左上角，高频分量集中在右下角。为了不会较大的影响图像质量同时又不会被低通滤波影响，设置了阈值 border，用来控制低频和高频中不进行数据隐藏的区域的大小。
2. 极值点之间会产生一定的干扰，解码时会产生错判和误判，设置了阈值 sparsefactor，用来控制两个极值点之间的最小距离。
3. 解码效率较低，因为要对每一个点进行极值判断。改进为对每一个点判断为极值后，将该点和 sparsefactor 所确定的区域内的所有点的标志位置 1，表示不用再对其进行极值判断。
4. 编码效率不高，每一点只能代表一个二进制位，若用 RGB 三层的有/没有极值作为信息，则可以表示三个二进制位。（待实现）

5. 使用最左上进行判断多边形链的起点有可能会被一些变换所影响，可以考虑改为当 RGB 三层同时存在极值时该点做为起点，并将逐层求凸包改为直接求一个左螺旋的单调凸链。（待实现）

[测试、对比结果]

同王弘宇毕设期间的实现相比，数据的隐藏量有了一定提高，因而极值点密度大幅度下降，同时由于正态分布中心密集而四周稀疏，所以隐藏后图像和原图像相比差异很小，一定程度上提高了质量和安全性。但同时由于采用的是凸包算法，识别结果同每个点位置的相关性都很大，一旦发生错判、漏判或者剪切攻击，对识别的影响将是致命的，可能会造成完全不正确的结果。其他对于旋转、错切、度量等变换的鲁棒性，经过后来的简单测试，大致和原方法近似，主要依赖于 DCT 变换的相对不变性，只能接受小范围之内的变化。

[没有解决的问题]

虽然对于极值点错判和漏判的问题通过 sparsefactor 的设置得到了一定的改善，但由于 DCT 反变换到 0-255 范围内量化和滤波的限制，仍然在个别具体的图像和随机数分布中会产生错判和漏判，还需要今后对各种图像的 DCT 变换系数进行进一步的分析和统计，以便找到一个更好的设定极值或者其他形式特征点的方法。

[有关文献及代码来源]

全部代码中除用 VC 的 Classwizard 生成的代码和直接调用 Matlab 生成的 C++ 函数（如 dct2 等）其他都是我们独立完成。

参考文献**GEOMETRIC-INVARIANT ROBUSTWATERMARKING THROUGH CONSTELLATION MATCHING IN THE FREQUENCY DOMAIN**，*R.Caldelli*a , *M.Barni*b , *F.Bartolini*a *and A.Piva*，**ICIP2000**。

[其他需要说明的问题]

1. 抗剪切攻击的对策：对于抗剪切攻击的要求可以将原图分为适当若干部分，每一部分的大小要足够隐藏待隐藏信息，每一部分分别作 DCT 变换，将待隐藏信息在每一部分的频域图中都进行隐藏，这样只要有一部分图像没有被剪切掉，信息就能被正确恢复出来。
2. 向密钥式算法的转化：实际上，每次生成随机数点集的随机种子或者随机函数，可以作为密钥式算法的密钥，这样解码方只要获得了这个密钥，就可以完全的重新生成一模一样的随机数序列，可以在相当短的时间内完全解码，而不必像现在这样对每点进行搜索。