实验五　基于STC编码的空域图像自适应隐写

姓名：王运韬　　**学号**：201628018627123

一、实验目的：

学习使用*DDE*实验室*STC*编码的*MATLAB*实现，使用*STC*编码[1][2]结合自适应隐写代价函数$S-UNIWARD$[3]对空域图像实施自适应隐写嵌入。

二、实验原理：

*STC*编码全称为*Syndrome Threllis Code*，即伴随式栅格编码，是一种专用于寻找自适应隐写路径的编码算法，衍生于信道编码中卷积码的解码算法[4]。其核心在于使用*Viterbi*算法寻找最小化嵌入代价和的嵌入路径，因此在使用该算法前需要对嵌入路径上的每一个像素计算嵌入修改代价。

本实验使用了$S-UNIWARD$[3]（空域*UNIWARD*）代价函数来计算图像像素的修改代价。在显示嵌入修改图案的图像文件*location.bmp*中，黑点的位置即被修改的像素位置，这些位置就是经过*STC*算法选择修改代价最小的位置。

*STC*编码对整个图像文件进行全局操作，无需对像素序列进行分组。

完成隐写嵌入后可以查看*location.bmp*对嵌入效果进行验证。由于*STC*编码是自适应方法，嵌入改变集中于图像的纹理区域及边缘区域，因此*location.bmp*可以勾勒出原图像的大体轮廓，可以以此为标准对自己编写的代码进行运行效果验证。

*STC*的一些性质：

1.若α为负载率，则$\alpha=1/w$，w为小矩阵的列数；

2.校验矩阵H每列最多有h个非零元素，y_i仅影响h个消息比特；

3.校验矩阵H的列数$n=m\cdot w=m/\alpha$，其尺寸可以表达为：

$$m\times n=[\alpha n]\times n=m\times(m\cdot w)$$；

4.由于α是小数而w是整数，不能获得任意的负载率，分别采用宽度w和$w+1$两个子矩阵，$1/(w+1)<\alpha<1/w$，可以通过混合排列子矩阵逼近α；

5. *STC* 的求解与路径优化过程可以通过格图来表达。在格图中，任何一个仍在发展的路径代表仍可能满足。

二元编码格图：

子块。格图包含 m 个子块，依次对应 $H_{m\times n}$ 中对角线方向相应的 $\hat{H}_{h\times w}$ 提取方程求解；第 i 个子图通过 w 层的发展确保了 m_i 可通过 $H\boldsymbol{y}=\boldsymbol{m}$ 提取， y_i 决定了是否将 H 的第 i 列加到校验子中；每个子块有 2^h 行与 $w+1$ 列，子块中两个相邻列上的节点由连线从左向右连接。

节点。每个子块节点表示一种阶段状态，连线通过的节点为可达节点，对应当前有效路径上的状态；在 2 个子块间，按照 $H\boldsymbol{y}=\boldsymbol{m}$ 的原则选择可达节点进入下一个子块的第一列，当前状态低位删除高位补 0，因此子块首列节点表示开始或者在上轮基础上开始新的子块计算。

位置状态（列编号）。除了每个子块的第一列，整个格图每列依次编号为 $\{1,2,...,n\}$，表示当前考虑修改的载体位置；第 i 个子块的首列用 p_i 表示。

局部校验子状态（行编号）。每行依次编号为 $\{0,1,...,2^h-1\}$，一般用二进制表示当前得到的局部校验子，局部校验子值的二进制是低位靠右，对应校验子向量的上部。

代价状态（节点标注）。节点上记录的数字表示编码代价，即至目前，前面修改积累的代价总和。

连线。子块内每个可达节点的 2 个分支代表不同的嵌入方法，即令 y_i 为1或 0；子块间连线表示有效状态节点进入下一子块，状态进入下一子块时当前状态低位删除高位补 0。连线的方向并不表示特定的修改方式，它仅连接 y_i 当前的状态以及不修改后或者修改后的状态。

若采用三元嵌入，如用随机 ±1 的方式进行修改，并且 $+1$ 与 -1 的单点代价被设计为相等的，则第一层 *LSB* 修改对 $+1$ 或 -1 并没有形成约束，可以通过控制 $+1$ 与 -1 操作次 *LSB* 层。

在第二层中（次 *LSB*），若将第一层修改位置上的单点代价标记为 0，其他点较大，则可以进行低代价的第二层嵌入：第二层代价为 0 的点若要修改，仅需控制第一层是 $+1$ 还是 -1，未引入新的能量扰动。

相比于 *STC* 编码，矩阵编码、*MME* 和湿纸编码再优化上的局限性：

矩阵编码仅仅减少修改次数，在分段中不参照载体样点的特性；

MME 在分段中参照了载体样点的特性，但是受到的制约较大；

湿纸编码实现了依据原始载体局部特性的动态位置嵌入，并且这种动态性不影响消息的正常接收。但是，湿纸编码较难构造，尤其是，修改位置选择缺乏优

化控制。在湿纸编码中，已经出现嵌入损失（*Embedding Distortion*）的概念，但损失是用“干”或“湿”来简单衡量的，在能够使提取方程满足的多种位置组合中，在干点范围内缺乏优化选择。

三、实验步骤：

1.构造子校验阵。在嵌入前需要根据需要构造子校验阵 $\hat{H}(H\_hat)$，使用 $\begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}$，在代码中表示为 $[3\ \ 2]$，嵌入率为 $\alpha = 1/2$。

2.生成嵌入消息序列。根据像素序列长度与嵌入率确定可嵌入信息的长度，根据消息序列长度生成随机 $0-1$ 序列作为消息序列。

3.构造校验矩阵。调用函数 *create _ code _ from _ submatrix* () 构造校验矩阵。

①函数 *create _ code _ from _ submatrix* () 参数说明如下：

h _ hat：构造校验矩阵的子矩阵；

num _ of _ sub _ blocks：校验矩阵中子矩阵的个数。

②函数 *create _ code _ from _ submatrix* () 返回值为：

code：包含校验矩阵信息的结构体，作为维特比算法的输入函数；

alpha：根据子矩阵计算的嵌入率（此返回值是结果值，对之后的计算没用）。

4.根据嵌入失真寻找最佳修改模式。调用函数 *dual _ viterbi*()，使用维特比算法根据各像素的修改代价寻找最佳的 *x* 的修改模式 *y*。

①函数 *dual _ viterbi*() 参数说明如下：

code：前一步骤计算出的包含校验矩阵信息的结构体，作为维特比算法的输入函数；

x：载体像素的 *LSB* 序列；

w：载体像素对应的嵌入失真序列（权重）；

m：需要嵌入的信息序列。

②函数 *dual _ viterbi*() 返回值说明如下：

y：修改（嵌入）之后的 *LSB* 序列；

min_cost：实施嵌入修改的代价（同样作为说明性能的结果值，不会作为其它函数的输入）。

四、实验结果

1.构造子校验阵。子校验阵 $\hat{H}(H\_hat)$，使用 $\begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}$，在代码中表示为 $[3\ \ 2]$，嵌入率为 $\alpha = 1/2$；

2.构造校验矩阵。 $code = creat\_from\_submatrix(H\_hat, imagezVecLeng)$

3.生成嵌入消息序列。根据像素序列长度与嵌入率确定可嵌入信息的长度，根据消息序列长度生成随机 $0-1$ 序列作为消息序列：

$$m = double(rand(sum(code.shift),1) < 0.5)$$

4.根据嵌入失真寻找最佳修改模。调用函数 $dual\_viterbi()$，使用维特比算法根据各像素的修改代价寻找最佳的 x 的修改模式 y。

下图所示为部分实验结果：

$min\_cost1 = 43793, imageVecLength1 = 262144, alpha = 0.5$

$min\_cost2 = 31469, imageVecLength2 = 188500, alpha = 0.5$

Fig 1. Cover Image

Fig 2. Stego Image

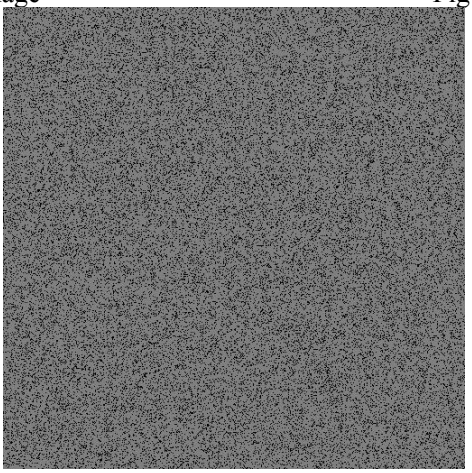

Fig 3. Embed pixel

Fig 4. Cover Image

Fig 5. Stego Image

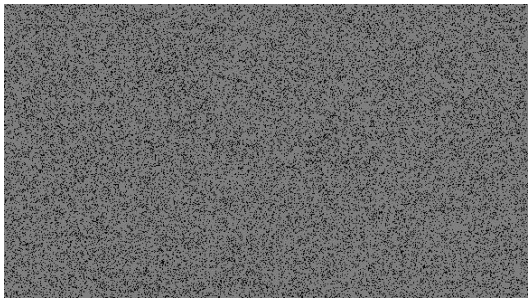

Fig 6. Embed pixel

BMP/JPG

Fig 7. Cover Image

图7不能完成*STC*隐写，由此我们可知*STC*编码适用于纹理信息较为丰富的图像，对图像形状无要求。

五、实验结论

*STC*编码是自适应方法，嵌入改变集中于图像的纹理区域及边缘区域，*embed pixel*可以勾勒出原图像的大体轮廓，可以以此标准对自己编写的代码进行运行效果验证。

*STC*编码对整个图像文件进行全局操作，无需对像素序列进行分组，其核心在于使用*Viterbi*算法寻找最小化嵌入代价和的嵌入路径，因此在使用该算法前需要对嵌入路径上的每一个像素计算嵌入修改代价，选择最小修改代价方案。

*STC*编码相对于矩阵编码、*MME*以及湿纸编码有更大的优越性和实用性，值得进一步研究和开发。

参考文献

[1].T. Filler, J. Judas, and J. Fridrich. Minimizing additive distortion in steganography using syndrome-trellis codes. IEEE Trans. Inf. Foren. Sec., 6(3):920-935, 2011.

[2].T. Filler, J. Judas, and J. Fridrich. Minimizing embedding impact in steganography using trellis-coded quantization. In Proceedings SPIE, Electronic Imaging, Security and Forensics of Multimedia XII, volume 7541, pages 1–14, 2010.

[3].V. Holub, J. Fridrich and T. Denemark. Universal distortion function for steganography in an arbitrary domain. EURASIP Journal on Information Security. Dec. 2014, 2014:1.

[4].V. Sidorenko and V. Zyablov. Decoding of convolutional codes using a syndrome trellis. IEEE Transactions on Information Theory. 40(5), 1663–1666, 1994.