Lecture 5’: The most effective methods of SG system design

1. HUGO project [63]

Problem definition

The goal of HUGO is to design stegosystem that is optimized to be as undetectable as possible against a particular algorithm of steganalysis.

In general – it is possible to optimize stegosystem against any stegoalgorithm.

HUGO is optimized against SPAM[62] – one of the best known algorithms of blind steganalysis.

After algorithm is chosen authors of HUGO decompose the problem into two parts:

1) Determine cost of changing of every pixel of an image

2)Perform the embedding with minimizing of total cost of all changed pixels across the image

Overview

Let X be cover image and Y is the image after embedding and $D(X,Y)$ a distortion function that shows how much X differs from Y in the sense of ability of SPAM stegoalgorithm to distinct X from Y. Then goal of HUGO embedding is to minimize value (1) while making extraction of embedded message possible:

$$D(X,Y) = \sum_{i=1}^{n_1 \times n_2} \rho_i |x_i - y_i| \qquad (1)$$

where

$0 \leq \rho_i \leq \infty$ - is the weight coefficient (the cost of changing i-th pixel of X),

x_i, y_i - values of *i*-th pixel of X and Y correspondingly

Changes are restricted to ±1 so that following equation holds:

$|x_i - y_i| \leq 1$

Additive form of (1) means that detectability of SG does not depend on correlation between embedded bits. That assumption holds when changed pixels are far from each other, which in turn holds when embedding rate is relatively low.

Formula (1) shows a general approach to stegosystem design and does not show how to construct a stegosystem that minimizes D(*X,Y*).

To design such stegosystem it is necessary to solve two problems:

1) How to adequately choose weights ρ_i so that minimization of (1) leads to improving undetectability ?
2) What coding method to use for changing pixels according to their weights ρ_i ?

Weights calculation

In order to get pixel weight for using in formula (1) we change pixels of X one by one and calculate "cost" of the change between X and $Y^{(i,j)}$

$$\rho_{i,j} = D(X, Y^{(i,j)}) \quad (2)$$

where (*i,j*) - are the coordinates of changed pixel.

Following formula is used to calculate weighted distortion between two images

$$D(X,Y) = \sum_{d_1,d_2,d_3=-T}^{T} \left[w(d_1,d_2,d_3) \left| \sum_{k\in\{\rightarrow,\leftarrow,\uparrow,\downarrow\}} \mathbf{C}^{\mathbf{X},k}_{d_1d_2d_3} - \mathbf{C}^{\mathbf{Y},k}_{d_1d_2d_3} \right| + \quad w(d_1,d_2,d_3) \left| \sum_{k\in\{\searrow,\nwarrow,\swarrow,\nearrow\}} \mathbf{C}^{\mathbf{X},k}_{d_1d_2d_3} - \mathbf{C}^{\mathbf{Y},k}_{d_1d_2d_3} \right| \right] \quad (3)$$

where features $\mathbf{C}^{\mathbf{X},k}_{d_1d_2d_3}$ and weight function $w(d_1,d_2,d_3)$ calculated as shown below ((4) and (5))

Let's consider one of the most effective blind steganalysis method – Subtractive Pixel Adjacency Matrix (SPAM) [62]

SPAM features are transition probabilities between pixels for eight directions {←,→,↑,↓,↖,↘,↗,↙} . Example for horizontal direction is given below:

$$\mathbf{C}^{\mathrm{X},\rightarrow}_{d_1d_2} = \Pr(\mathbf{D}^{\rightarrow}_{ij} = d_1, \mathbf{D}^{\rightarrow}_{i,j+1} = d_2), -T \le d_1 \le T, -T \le d_2 \le T \quad (4)$$

where $\mathbf{D}^{\rightarrow}_{ij} = \mathrm{X}_{ij} - X_{i,j+1}, 1 \le \mathrm{i} \le n_1, 1 \le \mathrm{j} \le \mathrm{n}_2 - 1,$
and I is image of size $n_1 \times n_2$

Total number of features is $8 \times (2T+1)^{\boldsymbol{d}}$, where *d* is dimension (in the example above *d*=2)

Features are not equally important for steganalysis. Authors of [58] suggest following function to weight features

$$w(d_1,d_2,d_3) = \frac{1}{\left[\sqrt{d_1^2+d_2^2+d_3^2}+\sigma\right]^{\gamma}} \quad (5)$$

where σ , γ – are parameters that can be changed to tune algorithm.

Syndrome-Trellis Codes

Problem definition:

Let $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$ be a binary vector of length n (in our case it contains LSB of cover image $\mathbf{X}$),

$\boldsymbol{\rho} = \{\rho_1, \rho_2, \dots, \rho_n\}$ is the vector of real values corresponding to weights of changing corresponding to changing of each bit of vector $\mathbf{x}$(in our case they are calculated according to formula (5))

$\mathbf{m} = \{m_0, m_2, \dots, m_{k-1}\}$ – binary chain of message bits of length k,

$\mathbf{y} = \{y, y_2, \dots, y_n\}$ - binary vector of length n

The goal is to encode message bits $\mathbf{m}$ into vector $\mathbf{x}$ so that distortion in sense of (1) is minimized.

Algorithm description:

Authors of [63] suggest to use trellis coding and Viterbi algorithm (VA) for minimization of D(*X,Y*) in general case i.e. $\rho_i \neq 1$, $\rho_i \neq \infty$.

Let $\mathbb{H}$,a binary matrix of size $k \times n$, be generating matrix for some code.

Then, the problem is to choose such $\mathbf{y}$ that following conditions holds:

$1) \mathbb{H}\mathbf{y} = \mathbf{m}$ (6)

2) Choose such solution of (6) that for given $\mathbf{x}$ and $\boldsymbol{\rho}$ - $D(\mathbf{x}, \mathbf{y})$is minimized.

In [63] it has been suggested to use special form of matrix $\mathbb{H}$ that is constructed from several copies of small submatrix $\hat{\mathbb{H}}$ of size $h \times w$ next to each other with one row shift. Next, for purpose of simplicity we will use matrix $\hat{\mathbb{H}} = \begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{pmatrix}$. Number of copies $\hat{\mathbb{H}}$ into $\mathbb{H}$ is equal to k – number of message bits.

$$\mathbb{H} = \begin{pmatrix} h_{11} & h_{12} & & & & \\ h_{21} & h_{22} & h_{11} & h_{12} & & \\ & & h_{21} & h_{22} & & \\ & & & & \cdot & \\ & & & & & \cdot \\ & & & & & & \cdot \end{pmatrix}$$

It can be easily seen that due to this form of matrix $\mathbb{H}$ each bit of $\mathbf{m}$ affected only by $2w$ bits of $\mathbf{y}$, and each bit of $\mathbf{y}$ affects only h message bits. This significantly reduces number of solutions of (6) and makes it possible to calculate solution of (6) step by step.

In coding theory, there is a definition of syndrome $\mathbf{s} = \mathbb{H}\mathbf{y}$,which corresponds to $\mathbf{m}$. During the encoding process one bit of partial syndrome is calculated on each step. Partial syndrome calculated on each step is of length h.

Trellis consists of $2^h \times k(w+1)$ nodes. Each of 2^h states correspond to bits of partial syndrome $(s_{i+h}, s_{i+h-1}, s_i)$ where $1 < i < k$. Let's consider first block of the trellis for matrix $\mathbb{H}$ given above. First block of such trellis is shown on the next slide.

As it was aforementioned, state for the first block correspond to partial syndrome (s_1, s_0). Taking into consideration that $\mathbf{s} = \mathbb{H}\mathbf{y}$, partial syndrome

$$\begin{pmatrix} s_0 \\ s_1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{pmatrix} \times \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \quad (7)$$

Columns marked as 1 and 2 correspond to the corresponding bits of $\mathbf{x}$, p_0 is dummy column that does not correspond to actual bit of $\mathbf{x}$ but is needed for purposes of state transition between blocks(because with shifting to next block bits of partial syndrome are always being shifted)

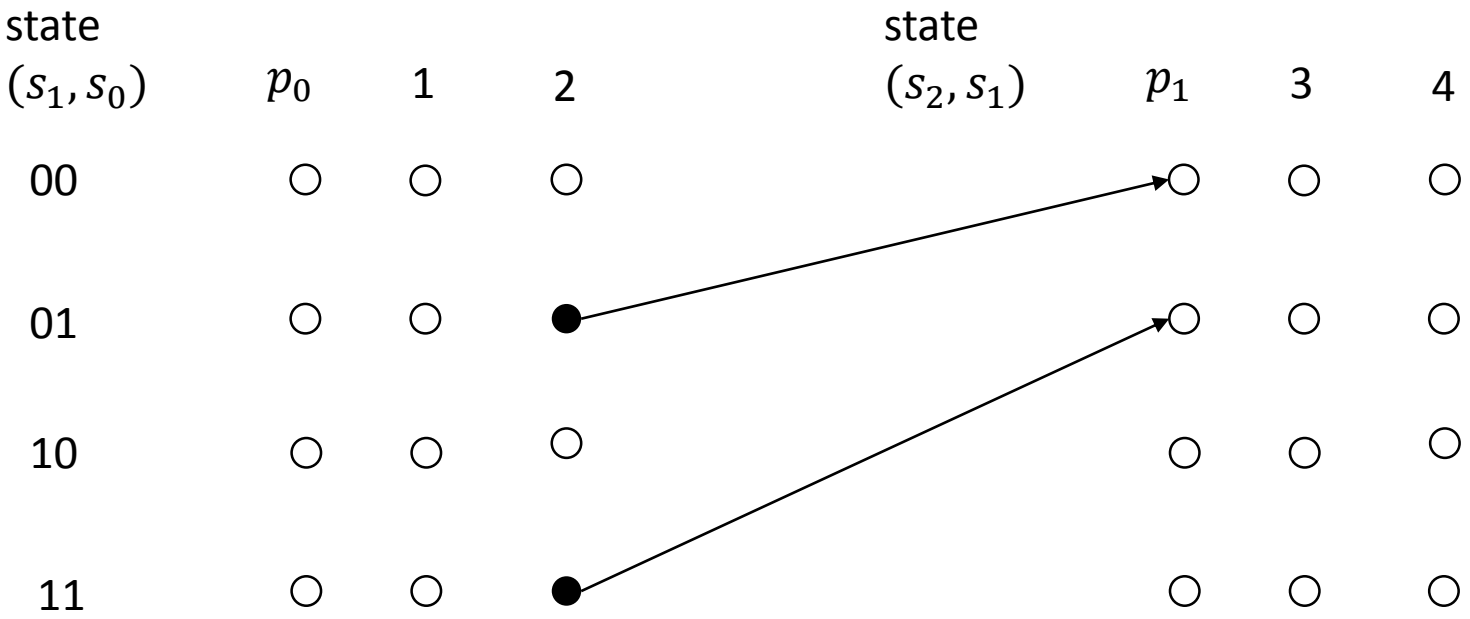
state (s_1, s_0)	p_0	1	2
00	●	○	○
01	○	○	○
10	○	○	○
11	○	○	○

At the beginning of the first block we start from state 00. Then from this state there are two path – corresponding to $y_1 = 0$ and $y_1 = 1$. Since we are performing step-by-step calculation of $\mathbf{s}$ and $\mathbf{y}$, on the first step we will use only first column of matrix $\mathbb{H}$(or we can say that we 'set' $y_2 = 0$)

1) path corresponding to $y_1 = 0$. According to (7) the path will lead into state $\begin{pmatrix} y_1 h_{12} \\ y_1 h_{22} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.
2) Similarly, path corresponding to $y_1 = 1$ will lead into state $\begin{pmatrix} y_1 h_{12} \\ y_1 h_{22} \end{pmatrix} = \begin{pmatrix} h_{12} \\ h_{22} \end{pmatrix}$

At the end of this step, there will be path entering two states in the column 1 of the trellis. The weight of this path is set to ρ_1 if $y_1 \neq x_1$ and 0 otherwise. Similarly to first step, we use only second column of matrix $\mathbb{H}$ and depending on value y_2 there are two paths from each of the two states in column 1. Weight of the resulting paths are calculated additively across the trellis.

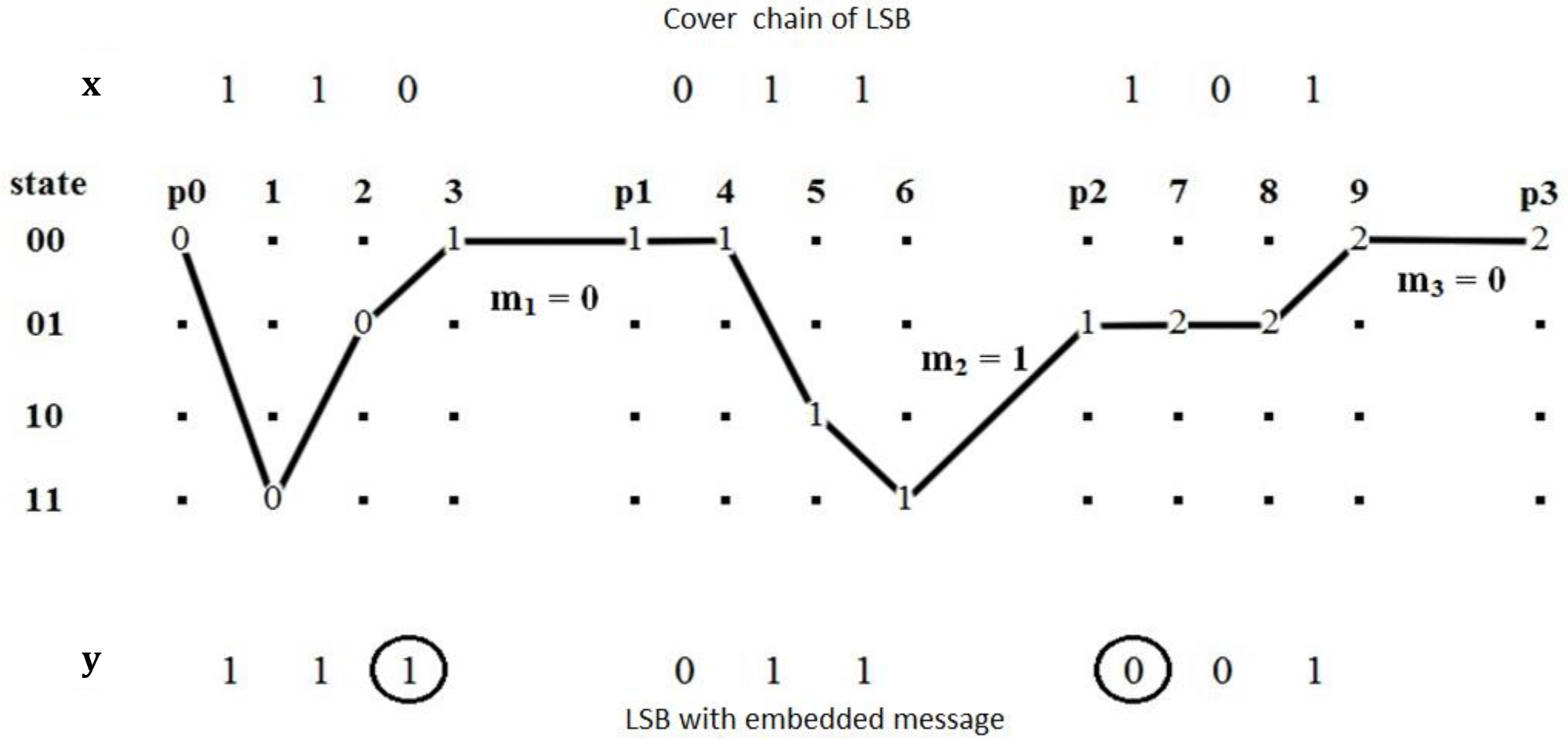
So, at the end of the first block there will be paths to every state of column 2. Remembering, that state corresponds to partial syndrome we choose only those states that have $s_0 = m_0$. As an example, let's choose $m_0 = 1$. The next step is transition from (s_1, s_0) bits of partial syndrome to (s_2, s_1) that is represented by transition to the next block. During transition, we leave s_0 out of partial syndrome, shift s_1 on it's place and set s_2 as 0.



Also, there are such situations when two paths enter one node in that case path with minimal weight is chosen as a survivor. Similarly, all bits of partial syndrome are calculated, and in the end of the trellis the only one path with minimal weight is chosen.

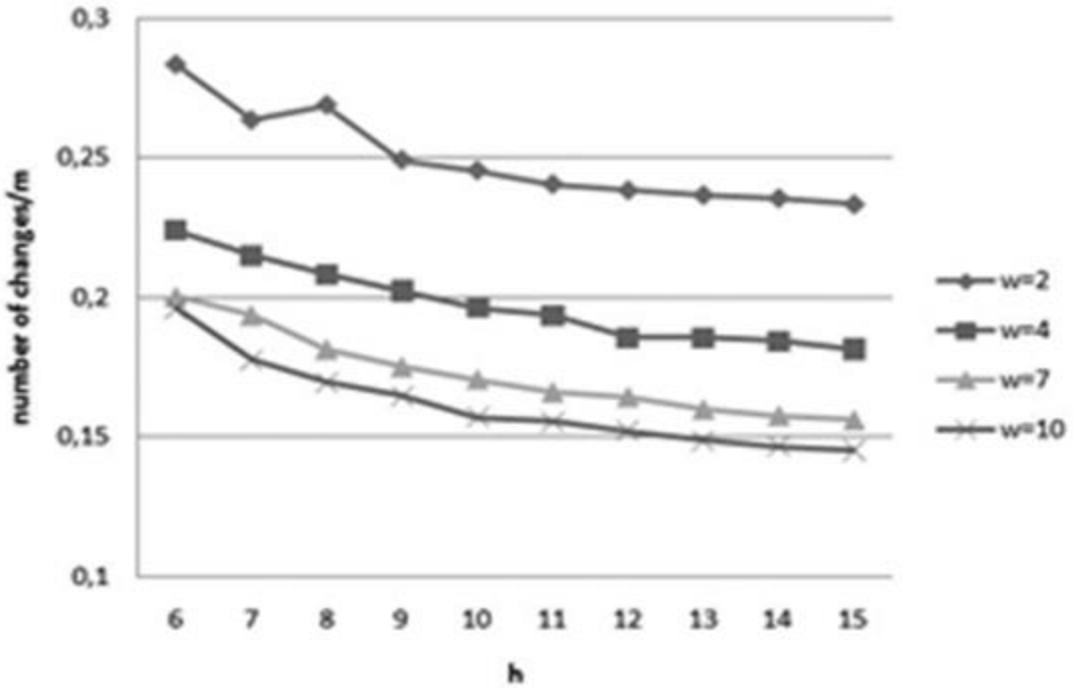
Example

Submatrix $\hat{\mathbb{H}} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$ of dimension size $h \times w$ = 2×3. LSB of CM are $\mathbf{x} = (1\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 1)$ and it is required to embed $\mathbf{m} = (0\ 1\ 0)$. Then, using method described in [63] we can build a path through the trellis (see figure below), which minimizes D(**x**,**y**) for particular case when ρ_i = 1, i = 1, 2, …, n.

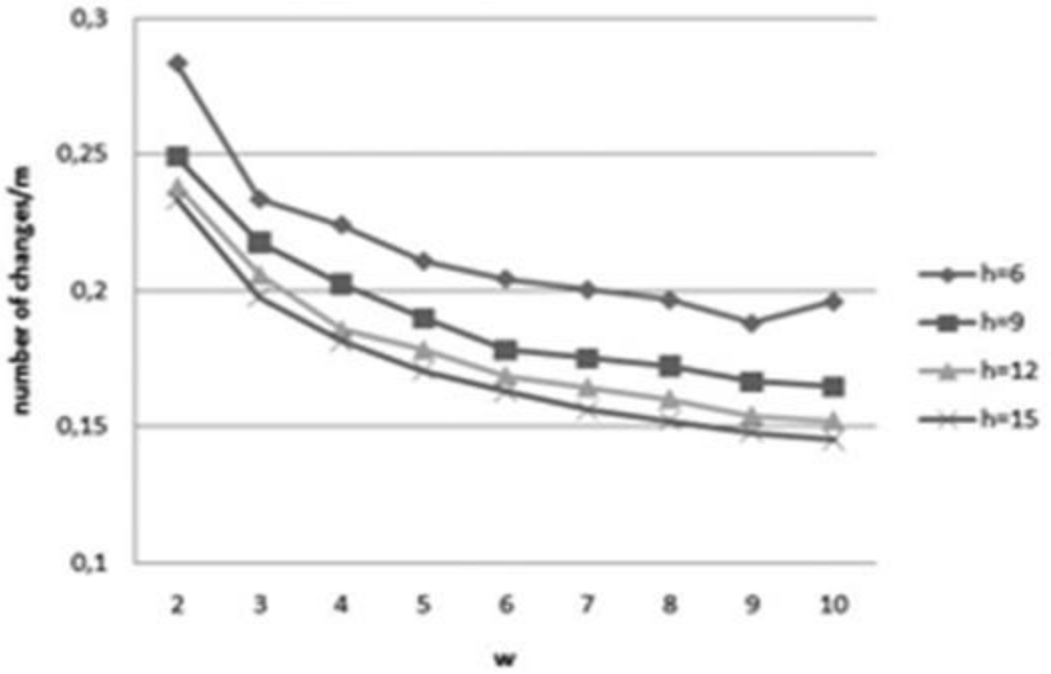


Experimental results

Dependency of relative number of changes from a) *h* and b)*w*



a



b

Comparing HUGO and LSB matching

LSB matching

Image size	Relative payload k/n	Probability of error during detection P_e
16x16	0.4	0.27
64x64	0.4	0.176
128x128	0.2	0.1780
128x128	0.4	0.123
256x256	0.4	0.099

HUGO

Image size	Relative payload k/n	T	σ	γ	P_e
16x16	0.4	10	10	4	0.337
64x64	0.4	10	10	4	0.24
128x128	0.4	10	10	4	0.162
128x128	0.2	10	10	4	0.269

2. Optimization of SG based on the use of nearest-neighbor approach to divergence estimation

Subjects of optimization are:

-methods of embedding

-parameters of SG algorithms

It has been proved in Lecture 5 that the value of **relative entropy** $D(X||Y)$ (or **Kullback-Leibler divergence**(**KLD**)) can be used in order to estimate efficiency of SG (in terms of P_m and P_{fa}) for the best detecting methods:

$$P_{fa}\log\left(\frac{P_{fa}}{1-P_m}\right)+\left(1-P_{fa}\right)\log\left(\frac{1-P_{fa}}{P_m}\right)\leq D(X||Y),$$

where $D(X||Y)=D(P_c||P_s)=\int P_c(z)\log\left(\frac{P_c}{P_s}\right)dz,$

P_c - the probability distribution of CO,

P_s - the probability distribution of stegosignal (SG).

Consider i.i.d. d-dimensional samples $X_1, \dots, X_n$ and $Y_1, \dots, Y_n$ drawn independently from densities *p* and *q* respectively.

If *p* corresponds to CO and *q* corresponds to SG, then it would be possible to estimate them given $\bar{X}, \bar{Y}$ and then to compute $D(\bar{X}||\bar{Y})$. But it is very time consuming process. Much better is to execute the nearest neighbor distance(NND) following to the paper[70].

At [70] the following NND estimator is introduced:

$$\overline{D}_{n,m}(p||q) = \frac{d}{n}\sum_{i=1}^{n} \log\left(\frac{\nu_m(i)}{\rho_n(i)}\right) + \log(\frac{m}{n-1})$$

Authors of [70] have proved the following relations:

$$\lim_{n,m\to\infty} E\{\widetilde{D}_{n,m}(p||q)\} = D(p||q)$$

$$\lim_{n,m\to\infty} E\{[\widetilde{D}_{n,m}(p||q) - D(p||q)]^2\} = 0$$

Application of NND estimator to steganography was proposed in [71].

Let us take one image and divide it into disjoint $(L \times L)$ – pixel areas chosen like a chess board, where white areas correspond to the set X (no embedding) and black areas correspond to the set Y (embedding by some algorithm).

Example of 12922×6080 pixels image is shown below:

$\alpha = 3$

The results of KLD calculations are present in Table o

p	LSB replcaing	LSB matching	SS		HUGO
			$\alpha = 1$	$\alpha = 3$	
0	-0.07	-0.07	-0.07	-0.07	-0.07
0.1	32.16	27.66	76.26	135.70	7.52
0.2	85.22	77.02	184.27	273.91	22.46
0.3	143.89	129.72	270.47	352.42	48.13
0.4	199.70	183.80	331.57	427.20	83.48
0.45	227.54	204.78	352.87	444.99	-
0.5	249.79	225.60	368.09	463.42	-
0.55	273.44	252.89	378.50	483.73	-
0.6	295.80	273.79	386.21	494.02	-
0.7	331.87	299.99	414.46	504.93	-
0.8	356.86	326.49	424.98	504.93	-
0.9	379.08	351.02	439.78	477.59	-
1	388.36	371.02	440.00	440.85	-

This experiment allows to make the following conclusions:

-no embedding results in a practically value $\overline{D}_{n,m}(p||q) \approx 0$ for all algorithms

-the greater is the embedding rate(the probability of embedding), the less is the SG security by $D(p||q)$ criteria

-LSB-matching is more secure than LSB-replacing

-HUGO is much more secure than all other SG

It is possible to optimize the structure of submatrix for HUGO algorithm.

In the Table below it is present a dependency between KLD and the structure of the 4×2 submatrix $\widehat{H}$ of the STC check matrix H:

Matrix $\widehat{\mathbf{H}}$	$\begin{pmatrix}1 & 1\\ 1 & 0\\ 0 & 1\\ 1 & 0\end{pmatrix}$	$\begin{pmatrix}1 & 0\\ 1 & 0\\ 1 & 1\\ 1 & 0\end{pmatrix}$	$\begin{pmatrix}1 & 1\\ 1 & 1\\ 0 & 1\\ 1 & 1\end{pmatrix}$	$\begin{pmatrix}1 & 0\\ 1 & 1\\ 1 & 1\\ 1 & 0\end{pmatrix}$
$\widehat{D}(x\|\|y)$	41.9	45.2	39.4	56.2

It can be seen that there exists an optimal submatrix, namely $\begin{pmatrix}1 & 1\\ 1 & 1\\ 0 & 1\\ 1 & 1\end{pmatrix}$, whereas the submatrix $\begin{pmatrix}1 & 0\\ 1 & 1\\ 1 & 1\\ 1 & 0\end{pmatrix}$ gives a much worse SG security.