

University of Malta
L-Università ta' Malta

CCE5208

Multimedia Security

Assignment 2
Detectability of Embedding

Daniel Bonanno
141295(M)
M.Sc. in ICT (Telecommunications)

Introduction

The aim of this assignment is to compare the detectability of two different embedding methods: LSB Replacement and LSB Matching with respect to embedding impact. This is done through the use of Support Vector Machines (SVMs) which are to be trained and tested by using the Subjective Pixel Adjacency Matrix (SPAM) features of images.

Question 1

Question a

Fridrich [1] defines the relative embedding capacity as follows:

$$\text{Relative Embedding Capacity} = \frac{\log_2 |M(x)|}{n}$$

where M is the set of possible messages, x is a possible cover image and n is the number of elements in x .

The message M may contain a maximum of αn bits, since we are restricted to embed a maximum of 1 bit per pixel in αn pixels in **both** LSB Replacement and LSB Matching. This results in $2^{\alpha n}$ possible messages, since a bit can be either a 0 or a 1. Thus, $\log_2 |M(x)|$ evaluates to $\log_2 |2^{\alpha n}| = \alpha n$.

Therefore the relative embedding capacity for both LSB Replacement and LSB Matching is evaluated to be:

$$\text{Relative Embedding Capacity} = \frac{\alpha n}{n} = \alpha \text{ bits per pixel}$$

The above statement makes sense, since α is defined to be the ratio of pixels which have data with respect to all the pixels in the image.

Question b

Since LSB Replacement and LSB Matching modify the cover image, distortion is introduced. Thus measurements such as the L_2 norm, can be used to determine the difference between the original cover image x and the stego-image y . The L_2 norm is defined as follows:

$$d_2(x, y) = \sum_{i=1}^n |x[i] - y[i]|^2$$

It is generally useful to express the distortion per unit pixel. Fridrich [1] notes that one popular measure to do so is the Mean Square Error (MSE):

$$MSE = \frac{d_2(x, y)}{n}$$

The above mentioned measures will now be discussed with respect to both LSB Replacement and LSB Matching.

LSB Embedding

Assuming a random message, a binary 1 and a binary 0 have equal probability of occurrence. This means that: $P(0) = P(1) = 0.5$.

Furthermore, it can be assumed that in an image, the least significant bit (LSB) of the pixels are either 0 or 1 with equal probability, meaning: $P_{LSB}(0) = P_{LSB}(1) = 0.5$. A similar assumption can be found in [1].

The table below demonstrates the possible scenarios with their respective probabilities:

Table 1: Possible Scenarios when Embedding the Data

Message Bit	LSB	LSB After Embedding	Change in LSB	Probability of this Event
0	0	0	No	$P(0) \times P_{LSB}(0) = 0.5 \times 0.5 = 0.25$
0	1	0	Yes	$P(0) \times P_{LSB}(1) = 0.5 \times 0.5 = 0.25$
1	0	1	Yes	$P(1) \times P_{LSB}(0) = 0.5 \times 0.5 = 0.25$
1	1	1	No	$P(1) \times P_{LSB}(1) = 0.5 \times 0.5 = 0.25$

From the above table, we can conclude that half ($0.25+0.25$) of LSBs will be modified. This means that, out of the αn pixels, 50% will have a change in their LSB, resulting in a delta of 1 in the pixel value. Thus the total changes in pixels is $\frac{\alpha n}{2}$.

The L_2 norm can be calculated as follows:

$$d_2(x, y) = \sum_{i=1}^n |x[i] - y[i]|^2$$

As discussed above, $\frac{\alpha n}{2}$ pixels will have no change and $\frac{\alpha n}{2}$ will have a change of magnitude 1.

$$\therefore d_2(x, y) = \frac{\alpha n}{2} \times 1 = \frac{\alpha n}{2}$$

The MSE can then be evaluated as follows:

$$MSE = \frac{\frac{\alpha n}{2}}{n} = \frac{\alpha}{2}$$

LSB Matching

Similar to LSB Replacement, in LSB Matching 50% of the pixels will not register a change in value whilst 50% of the pixels will have a change in their value by ± 1 , as can be seen from Table 1.

Due to the squaring when calculating the L_2 norm,

$$d_{2LSB\ Replacement} = d_{2LSB\ Matching} = \frac{\alpha n}{2}$$

$$\Rightarrow MSE_{LSB\ Replacement} = MSE_{LSB\ Matching} = \frac{\alpha}{2}$$

Therefore, it can be concluded that the expected relative distortion is equal for both embedding methods.

Question c

[1] defines the average number of bits that can be embedded per unit distortion using the following equation:

$$e = \frac{E_x[\log_2 |M(x)|]}{E_{x,m}[d_2(x,y)]}$$

From question (a), it was determined that $E_x[\log_2 |M(x)|]$ is αn for both LSB Replacement and LSB Embedding. Similarly, from question (b), $E_{x,m}[d_2(x,y)]$ is equal to $\frac{\alpha n}{2}$ for both algorithms. In both cases, this means that:

$$e = \frac{\alpha n}{\frac{\alpha n}{2}} = 2$$

Meaning that for every unit distortion, 2 bits can be embedded. This result makes sense: on average 50% of LSBs will be changed by both algorithms. Thus on average, for every bit embedded, there will be a distortion of $0.5 \times 1 = 0.5$ (since every bit change results in a difference of 1 from the previous value, as shown in previous question). Therefore embedding 2 bits should results in a distortion value of 1.

Furthermore, it can be observed that this value is independent of how much is embedded in the image, that is, this value is not dependent on α . Not only is this revealed from the equation, but this is expected since e can be used to measure an algorithm's embedding efficiency, which should be independent on how much is embedded. Essentially, it is a normalized measure. Another way to look at it is through the reciprocal: How much will an image be distorted by embedding 1 bit? This further emphasizes the fact that this measure is not dependent on α .

Question d

LSB Replacement should be more detectable than LSB Matching. Although they both have the same embedding distortion, in LSB Replacement we have asymmetry:

- even values are never decreased – their LSB is always 0, so it either remains the same or it changes to 1, resulting in an increase in the pixel value by 1.
- odd values are never increased – their LSB is always 1, so it either remains the same or it changes to 0, resulting in a decrease in the pixel value by 1.

This asymmetry is a bias which makes LSB Replacement more detectable. In fact, it causes artefacts in the histogram: for a fully embedded image (that is, $\alpha = 1$), a pair of adjacent even and odd bins tend to even out. Such an artefact can be seen in Figure 1, obtained from [1], where a staircase effect can be clearly observed. This aids in the detectability of LSB replacement, since it can be exploited by the histogram attack, as defined in [1].

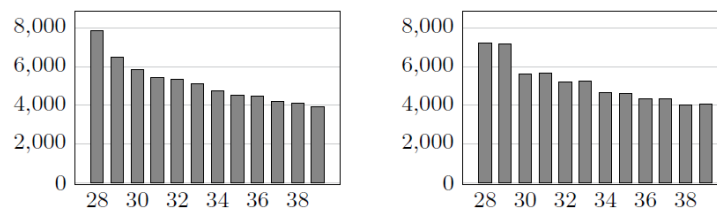


Figure 1: Zoomed in histogram of an image's intensities before and after embedding (left and right respectively) [1]

To remedy this, LSB Matching uses a method that is symmetric when embedding takes place – if the LSB needs to be changed, the pixel's value is randomly increased or decreased (except for 0 and 255, which are always increased or decreased respectively). It can be noted that this method means that it is possible that more than 1 bit is modified in the embedding process. However, the LSB is still equal to the embedded data. The staircase artefact is not present in the histogram as well. In general, this symmetric approach makes LSB Matching more robust.

Question e

Detectability in steganography is generally based on statistical analyses. It was observed in question (b) above that, for both LSB Replacement and LSB Matching, the distortion introduced in the image is directly affected by the embedding proportion α : the greater the value of α , the larger the distortion. Such distortions generally disrupt the normal statistics of an image. Thus, it is clear that, as α increases, detectability will also increase, both for LSB Replacement and LSB Matching. In fact, statistical restoration methods are sometimes used to try and get back the original statistics of the cover image. This is done by modifying the values of the pixels in the image which are not used in the embedding process.

A clear example of this is Figure 1. Although the staircase artefact in the histogram is observable only for LSB Replacement, it is intuitive that the more pixel intensities that are modified in the embedding process, the more pronounced the artefact in the histogram. As [1] points out, the histogram attack can identify images which are fully embedded (that is, $\alpha = 1$) but is only able to detect stego images with $\alpha < 1$ if the embedding path is known. This is further proof that α does affect the detectability. Although this artefact is not present for LSB Matching, a similar increase in distortion to other features will be present as α increases.

Question 2

Question a

The following python scripts were created:

- i. `Generate_Data_File.py` – Generate a data file having length based on the α parameter provided.
- ii. `LSB_Replacement.py` – Perform LSB replacement using the cover image and data file provided. The embedding is done for the alpha parameter given, ensuring that the data file is not too long. Furthermore, a key is used to enable extraction, since when embedding is not done in all pixels, the pixels are selected at random using this key.
- iii. `LSB_Matching.py` – Similar to `LSB_Replacement.py`. However, this script performs LSB Matching instead.
- iv. `LSB_Extraction.py` – This script takes as input the stego image and the key and extracts the data from the image, storing it in an output file.

Every script checks the input parameters provided, as will be discussed in the next section. More information on how these scripts were implemented can be found in the comments in the scripts themselves.

Question b

This section deals with testing the scripts that were implemented in the previous section.

I. Generate Data File

The first test carried out checks if the user has called the script with the correct amount of input parameters. If this is not the case, the script notifies the user, as shown in Figure 2, and exits.

```
daniel@ubuntu:~/Documents/Multimedia Security/Assignment 2/Question 2 New$ python Generate_Data_File.py
'test.txt'
Incorrect number of input arguments.
2 arguments must be input: the output file name and the alpha required
```

Figure 2: Incorrect number of input parameters

The next test ensures that the alpha parameter input is actually a numeric value. When this is not the case, the script notifies the user and exits, as shown in Figure 3.

```
daniel@ubuntu:~/Documents/Multimedia Security/Assignment 2/Question 2 New$ python Generate_Data_File.py
'test.txt' test
Please enter a number for alpha
```

Figure 3: Alpha is non-numeric

Alpha must be in between 0 and 1, with 0 being excluded and 1 being included. Thus, the next test ensured that the script notifies the user when alpha is not in this range, as shown in Figure 4 below.

```
daniel@ubuntu:~/Documents/Multimedia Security/Assignment 2/Question 2 New$ python Generate_Data_File.py
'test.txt' 1.1
Alpha must be less than or equal to 1
daniel@ubuntu:~/Documents/Multimedia Security/Assignment 2/Question 2 New$ python Generate_Data_File.py
'test.txt' -0.2
Alpha must be greater than 0
```

Figure 4: Alpha is out of range

Furthermore, it was ensured that the script would give a file having the correct number of bits for different values of alpha.

II. LSB Replacement and LSB Matching

Firstly, tests were conducted to ensure that the scripts would act accordingly when the number of input parameters are wrong, when alpha and the key provided are non-numeric and when alpha is out of range. The results are similar to those obtained for the Generate Data File script, as shown above.

Another test ensured that the scripts would notify the user if the cover image provided or the data file provided do not exist. The results can be found in Figure 5 below.

```
daniel@ubuntu:~/Documents/Multimedia Security/Assignment 2/Question 2 New$ python LSB_Replacement.py
'does_not_exist.pgm' 'Black_stego.pgm' 'test.txt' 1 1
cover image does not exist
daniel@ubuntu:~/Documents/Multimedia Security/Assignment 2/Question 2 New$ python LSB_Replacement.py
'Black.pgm' 'Black_stego.pgm' 'doesnotexist.txt' 1 1
Data file does not exist
```

Figure 5: Cover image or Data file do not exist

The scripts also ensure that the message length is not larger than the capacity defined by the alpha parameter provided. If this is the case, the user is notified, as shown in Figure 6.

```
daniel@ubuntu:~/Documents/Multimedia Security/Assignment 2/Question 2 New$ python LSB_Replacement.py
'Black.pgm' 'Black_stego.pgm' 'test.txt' 0.1 1
Data is too large
```

Figure 6: Message length larger than capacity

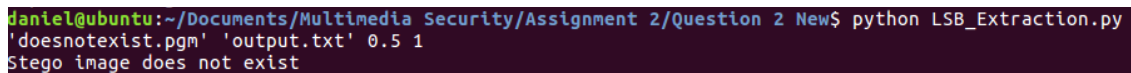
To ensure that the algorithms work correctly, they were tested on 1000 cover images from the BOSSbase image database [2], for alpha values in the range 0.1 to 1, with a step of 0.1. The choice of this dataset will be discussed further in Question 3. For every image and alpha combination, a data file and stego image were created and extracted, using the `Generate_Data_File.py` script and the `LSB_Extaction.py` scripts. The input data file and the output data file were then compared using the `diff` command in UNIX. For every case, the input and output file were equal, meaning that the embedding and extraction algorithms work correctly. The testing was done via an automated bash script called `Testing.sh`.

Furthermore, it is important to note that LSB Matching must cater for 2 corner cases: one where the pixel value is 0 and the other when the pixel value is 255. In these cases, the LSB Matching algorithm must only add in the first case and subtract in the second. Thus, to ensure correct operation, totally black and totally white 512×512 images were generated. They were then used as cover images using LSB Matching. If the algorithm works correctly, the black image should remain black, with no white speckles, since the white speckles would only be generated if 1 was subtracted from 0 and an overflow would occur. Similarly, there should be no black speckles in the stego image generated by the white cover image. This was indeed the case for both cover images, as can be observed from the images in the Question 2/Testing folder.

III. LSB Extraction

As previously mentioned, the LSB Extraction script was tested extensively using the `Testing.sh` bash script. Furthermore, it was tested for when the user inputs an incorrect number of input parameters, when alpha and the key are non-numeric and then alpha is out of range. The results are similar to previous those observed in previous sections.

A test was also carried out to check the output of the script when the provided stego image does not exist. The script notifies the user of this issue and exits, as can be seen in Figure 7.



```
daniel@ubuntu:~/Documents/Multimedia Security/Assignment 2/Question 2 New$ python LSB_Extraction.py  
'doesnotexist.pgm' 'output.txt' 0.5 1  
Stego image does not exist
```

Figure 7: Stego image does not exist

Another test was run to check the output of the script when the wrong key or the wrong alpha value are given. When the key is wrong, the extracted data file is different from the one that was embedded, as expected, since the sequence of pixels from which the LSB is extracted is out of sync with that of the embedding algorithm. The 2 files were compared using the `diff` function. When the alpha parameter is wrong, this results in less data being extracted if alpha is lower than that used in the embedding process or in more data being extracted if it is larger, assuming that the key is correct.

Question 3

Question a

As already mentioned, the training set used in this assignment is the BOSSBase image database, obtained from [2]. As noted in [3], this dataset has 10,000 never-compressed, greyscale, 512×512 cover images in PGM format, obtained from seven different cameras. This ensures that there is no bias due to some feature in the cover images obtained from the camera. Such a bias could be used by the steganalyzer to obtain better detection performances.

It was decided that, from the images in the dataset, 80% should be used for training and 20% used for testing, with no overlap between the training and testing set. This is a rule of thumb based on the Pareto principle. Lyu et al in [4] also use a similar training/testing split for their SVM, which makes use of colour wavelet statistics for steganalysis.

Question b

In this section, a bash script called `Get_Stego_Images.sh` was developed to obtain stego images for all the cover images found in the dataset described in (a). Stego images were created for alpha values in the set {0.1, 0.3, 0.5, 0.7, 0.9, 1} for both LSB Replacement and LSB Matching. The range of alphas chosen should give a general trend of how the detectability varies with respect to alpha. The developed script generates the hierarchy similar to the one shown in Figure 8, where the `LSB_Replacement` and `LSB_Matching` folders contain the stego images.

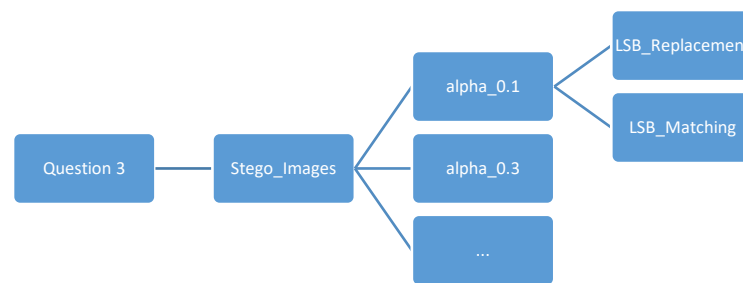


Figure 8: Hierarchy developed by the script

Question c

The techniques used to embed data aim to hide the data as noise in the image. However, Pevný et al [5] point out that dependencies in noise are present due to in-camera processing, creating opportunities for steganalysis techniques to exploit such dependencies between adjacent pixels. Rather than using the actual pixel values, however, the difference between adjacent pixels is calculated. This is done such that the model which must be observed is simpler in terms of complexity and to ensure that such a model is not dependent on the image content. This can therefore be seen as a form of calibration. In this experiment, the Subjective Pixel Adjacency Model (SPAM) is used to extract the features of each image along the 8 possible directions per pixel. These features are formed by Markov Chains. In this case, 1st order and 2nd order Markov Chains are utilized. Furthermore, the parameter T is used to restrict the range of difference value used in calculating the features. This can be done since most differences have low magnitude. In this experiment, T was set to 4 for the 1st order features and 3 for the 2nd order features, as specified. This leads to 162 1st order features and 686 second order features, as described in [5].

An implementation of how to extract the SPAM features was obtained from [6]. The script `Get_Spam_Features.sh` was created which makes use of the `spam.exe` programme from these tools to get SPAM features using the parameters provided for both the cover images and all the stego images obtained in the previous question. The hierarchy generated is similar to that found in Figure 8, with the root folder being `SPAM_Features`. It can be noted that 2 sets of files are generated: one for the 1st order features and another for the 2nd order features.

Question d

In this section, the data is being prepared to be fed into the SVM so that a classifier can be created. Following the previous section, the `Get_Training_Testing_Sets.sh` script must be run. First, this script calls the `Combined_1stOrder2ndOrder_SPAM_features.py` script to combine the 1st order and 2nd order feature files generated in question (c). This must be done since the LibSVM implementation, obtained from [7], has an executable called `fea2libsvm` which must be used. This programme serves 2 purposes:

1. Convert the features into a format that is suitable for the SVM implementation found in LibSVM
2. Split the data into training and testing, according to the ratio provided

In question (a), the decision was made to split the data into 80% training and 20% testing. The data is split randomly so as not to have any bias. Furthermore, an equal number of cover and stego images are used in both training and testing sets, so that one class is not favoured over another.

After `fea2libsvm` is run on the cover and stego image features, the training and testing features are scaled to the range -1 to 1, so that features with large ranges do not dominate features which have a smaller range, as noted in both [8] and [9]. In fact, in [9], there are several examples which show that scaling improves the classification rate.

The `fea2libsvm` programme includes both cover images and their stego counterparts in both the training and the testing sets. However, it can be argued that training an SVM using both the cover and stego images may over-fit the classifier to the training set, resulting in worse performance on the testing set. Furthermore, if the cover and stego versions of the same image are used in training, a bias in the classification results may be present. Assume an image is misclassified due to the content of the image, rather than due to the fact that data has or has not been embedded. This would automatically mean that one of the results will be correctly classified due to the content if both cover and stego versions of the image are used, leading to a detection result which does not truly reflect the performance of the classifier. Therefore, to avoid this scenario, the `Get_Training_Testing_Sets.sh` script calls the `Different_Cover_Stego.py` script to ensure that the cover and stego versions of the same image are not used in the same training or test set. This script randomly chooses which cover and stego images are kept whilst ensuring that amount of cover images present in one set is equal to the amount of stego images present in the same set. Again, a hierarchy similar to that shown in Figure 8 is created, with the root folder being `Training_Testing_Features`.

Question e

An SVM Classifier was trained and tested for every alpha value for both LSB Replacement and LSB Matching. This was done using the above training and testing sets by using the `svm-train` and `svm-predict` executables from the LibSVM [7] implementation. To do this, the `SVM_Train_Test.sh` script is utilised. It is important to note that sometimes, features are not able to be linearly separable. Hence, a kernel must be utilised to map the data into a higher dimensional space. To this aim, the RBF kernel was used, which requires parameters C and γ . It is not trivial to choose the optimal values for C and γ , which is why a grid search may be used, which trains and tests the classifier for a range of C and γ values. Since this is done using only the training set, cross-validation is used, which aims to avoid overfitting. Cross-validation splits the training set into k equal sized sets. Training is run on $k-1$ of those sets and validation is done using the last available set. This is repeated for k times, after which the validation accuracies are averaged out. The LibSVM implementation has a script,

grid.py, which performs this grid search. This script is used in the SVM_Train_Test.sh to obtain the optimal C and γ values, before training the classifier using these optimal values over the whole training set. The grid.py script performs 5 fold cross validation. It does so for values of C in the range 2^{-5} to 2^{15} , using a step size of 2^2 and for values of γ in the range 2^3 down to 2^{-15} using a step of 2^{-2} . A contour plot, similar to the one in Figure 9, is created for every SVM, which shows how the accuracy varies with the combination of parameters.

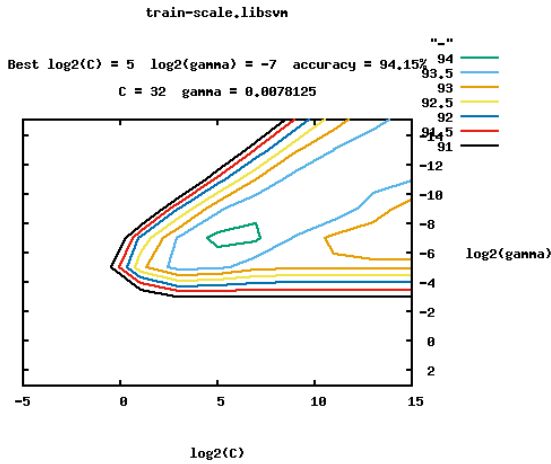


Figure 9: An example of the contour plot generated to find the optimal C and γ values

Again, the script generates a hierarchy similar to that of Figure 8, with the root folder being SVMs. Table 2 below shows the best C and γ values, as well as the results for every alpha used for LSB Replacement. Similarly, Table 3 shows these details for LSB Matching. The graph in Figure 10 shows how the accuracy varies with α for both LSB Replacement and LSB Matching.

Table 2: LSB Replacement - SVM Details

LSB Replacement			
Alpha	C (2^\wedge)	Gamma (2^\wedge)	SVM Detection Accuracy (%)
0.1	11	-11	88.2
0.3	5	-9	97.35
0.5	7	-11	99.15
0.7	7	-9	99.45
0.9	15	-15	99.7
1	9	-11	99.75

Table 3: LSB Matching - SVM Details

LSB Matching			
Alpha	C (2^\wedge)	Gamma (2^\wedge)	SVM Detection Accuracy (%)
0.1	7	-9	87.35
0.3	5	-7	94.6
0.5	7	-7	96.5
0.7	7	-7	97.65
0.9	5	-7	98.35
1	5	-5	98.9

Question f

For both algorithms, the detection accuracy increased as alpha increased. This was expected, as discussed in question 1e, since the larger the value of alpha, the larger the distortion in the image, resulting in larger discrepancies in the 1st order and 2nd order statistics. Furthermore, it is evident from Figure 10 that LSB Matching is slightly less detectable than LSB Replacement. This means that the asymmetry introduced by LSB Replacement, as discussed above in question 1d, does have an effect on the SPAM features. However, it can be noted that at the extremes, that is $\alpha = 0.1$ and $\alpha = 1$, the discrepancy is much less than for the middle values. With all this in mind, the SPAM features have proven to be quite effective when it comes to classification of whether an image is a stego image or not if it has been compressed with LSB Replacement or LSB Matching, with the lowest detectability rate being approximately 87%.

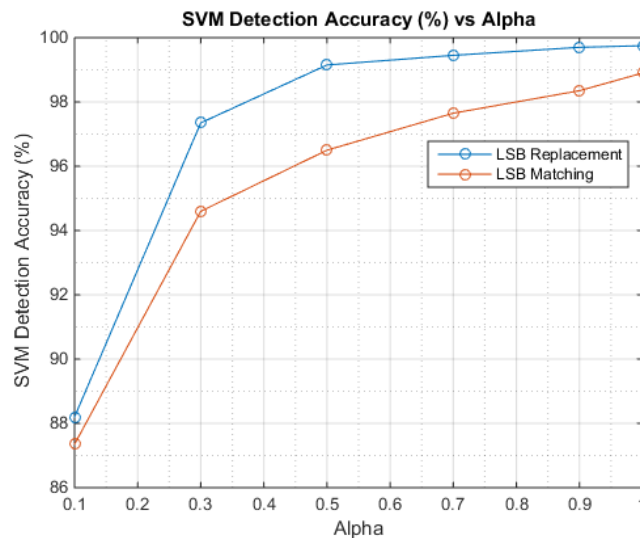


Figure 10: SVM Detection Accuracy vs Alpha for LSB Replacement and LSB Matching

References

- [1] J. Fridrich, *Steganography in Digital Media, Principles, Algorithms and Applications*, 1st ed. Cambridge: University Press, 2010.
- [2] "BOSSbase Image Database" [Online]. Available: <http://agents.fel.cvut.cz/stegodata/PGMs/>. [Accessed: 7-May-2017]
- [3] P. Bas, T. Filler and T. Pevný, "Break Our Steganographic System": The Ins and Outs of Organizing BOSS," 2011, pp. 59–70.
- [4] S. Lyu and H. Farid, "Steganalysis using color wavelet statistics and one-class support vector machines," 2004, p. 35.
- [5] T. Pevny, P. Bas and J. Fridrich, "Steganalysis by Subtractive Pixel Adjacency Matrix", *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 2, pp. 215-224, 2010.
- [6] "Implementations of HUGO embedding simulation, SPAM features extraction and tools for using with LibSVM, for Unix-based systems" [Online]. Available: <https://jabriffa.wordpress.com/other/hugo-source-code/>. [Accessed: 7-May-2017]
- [7] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1--27:27, 2011. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>. [Accessed: 7-May-2017]
- [8] H. G. Schaathun, *Machine Learning in Image Steganalysis*. Chichester, UK: John Wiley & Sons, Ltd, 2012.
- [9] C. J. Lin, C.W. Hsu and C. C. Chang, "A Practical Guide to Support Vector Classification," *BJU Int.*, vol. 101, no. 1, pp. 1396–400, 2008.