FINAL PROJECT

(WEIGHT 2, DUE BY MIDNIGHT OF DECEMBER 13)

Implement your own stego method and test its security

In this project, you get to implement your own steganographic method for the spatial domain and then test its security empirically using a database of 1000 images using an ensemble classifier and the KB feature vector.

1. Step 1 – Prepare cover images

Download 1000 512×512 8-bit grayscale images from Blackboard. They are in the 165MB archive 'covers.zip'. Unzip them all into a folder on your computer – it will contain 1000 pgm images for your tests.

Also, create six other folders where you will be storing your stego images embedded with payloads 0.05, 0.1, ..., 0.5 bpp. You might call them, for example, stego005, stego010, ..., stego050 or whatever you like. Each will require about 165MB.

2. Step 2 - Define your costs

Write a Matlab routine that computes the pixel costs for a given input 8-bit grayscale image. General advice for good results: make the costs small in textured regions of the image and large in smooth regions. You could use the cost assignment from the assignment on RD bound if you are absolutely lost. One caveat, though – make sure the output of your cost function assigns costs to ALL pixels in the image. The costs from the assignment on RD bound only covered the pixels in the inner 510×510 region. It is important that your costs are assigned to all pixels (in other words, the cost matrix should have the same dimension as the input image). Avoid costs that might come out as Inf or NaN in Matlab.

3. Step 3 - Compute your stego images for six payloads

Use the supplied 'simulate_emb.m' function to simulate embedding with *your* costs on the rate–distortion bound. Notice that this function accepts the following three inputs:

(1) Cover image X
(2) Relative payload α in bits per pixel (bpp)
(3) Embedding costs ρ (must be of the same dimension as X)

The output is a stego image Y that "contains" the relative message of length α. Remember, this is only a simulator and no actual embedding occurs. The function merely changes the pixels by ± 1 (LSB matching) based on the embedding probabilities $\beta_i = \exp(-\lambda\rho_i)/(1+\exp(-\lambda\rho_i))$, where the parameter λ is computed inside the function using a binary search. Inspect the code to see how it works. This will be useful for you from the learning perspective.

Write a function that reads a cover image from your cover folder, "embeds" a message using 'simulate_emb.m', and stores it under the same name (I recommend you use the pgm format again) in the corresponding stego folder (the folder with the corresponding payload). Friendly advice – make sure you cast Y to uint8 before saving using imwrite.m. If you have enough space on your disk, you might want to do all six payloads at the same time in one routine. Or you can do it one payload at a time. Use code fractions from 'hint.m' on how to read names of image files from a folder (basically use the function 'filenames.m') and then loop over the images.

At this point, you should have the following on your computer: a cover folder with 1000 cover images and six stego folders, each with 1000 stego images "embedded" with payloads 0.05, 0.1, ..., 0.5 bpp. You are now ready to steganalyze your embedding scheme.

4. Step 4 - Compute the KB features for all cover and stego images

Use the code fraction 'hint.m' to write a routine that does the following: reads an image from a specified folder, extracts its KB feature, and then saves all the symmetrized features (in a big 2D matrix) and file names (in a cell array) in one mat file. Do not worry, the 'hint.m' contains all the info and code you need.

Run your function for the folder of covers and then for each stego folder. I recommend that you store the output mat files in the corresponding folder where your images, whose features you are computing, reside. At the end of this step, you will have one mat file computed for covers and six mat files computed for images from each stego folder. You are ready to run the ensemble.

5. Step 5 - Steganalyze your stego method using ensemble classifier

I pretty much give you the entire code for this in 'running_ensemble_example.m'. There you will see a loop over ten seeds used for splitting the features into random halves. For the ensemble, you need to specify the path to the cover features and to the stego features (for the specific payload). The ensemble will do Nruns = 10 random splits of the feature files to compute an estimate of the testing error and output the average (over ten splits) P_{E} error:

$$P_{\mathrm{E}} = \min_{P_{\mathrm{FA}}} \frac{1}{2}(P_{\mathrm{FA}} + P_{\mathrm{MD}}) = \min_{P_{\mathrm{FA}}} \frac{1}{2}(P_{\mathrm{FA}} + 1 - P_{\mathrm{D}}).$$

Repeat this for each payload and store the P_{E} and its statistical spread – average (mean) absolute deviation MAD.

The result of your monumental effort should be a graph showing $P_{\mathrm{E}}(\alpha)$ as a function of the relative payload, α. It should be a decreasing function of α. Include this plot with all your code for the final submission. Also include a description of the costs in a mathematical form so that I do not have to decipher it from your Matlab code.

GOOD LUCK!!