**Technical Report**

**Technical Report**

**Stegnography final version**

|  |  |
| --- | --- |
| Prepared by: | Eng. Mohammad Sakka |
|  |  |
| Date: | 4/1/2022 |
|  |  |

**REPORT SENSITIVITY**

Does the report have any of the following sensitivities?

Intended for journal publication YES

Results are incomplete NO

Commercial/IP concerns NO

Project Workflow:

* First we coded the approach that is illustrated in the “techniqal report 1.01” word document.
* But with modifing of crossover pseudocode to be like the following:

1. The structure of the chromosome
2. M1M2M3 ….MN
3. The problem we are trying to solve is to build a function named f that receives two chromosomes A, B and generate two chromosomes C, D where each of C, D contains information from A and B
4. [C,D]=F(A,B)
5. Assuming that A is combined of metameric Ai ,i=1…length(A)
6. And B is combined of metameric Bj , j=1 …length(B)
7. Assuming that we have a function called decompose
8. [Ai]=Decompose(A)
9. [Bj]=Decompose(B)
10. Let assume that we selected randomly two metameric one from Ai, a and one from Bj, b
11. There are the following cases to be discussed
12. (we define this as a and b do not share the same block)
13. Then a and b can be part of the new child
15. we avoid adding them to the same child
16. let us assume that have another function classify
17. [shared,non-shared]=classify(Ai,Bj)
18. Child =[]
19. While (not (the size of the message is completed))
20. we selected randomly two metameric from non-shared
21. remove them from the non-shared
22. add them to the child
23. if (the non-shared is empty)
24. break
25. end
26. end
27. while(the message is not completed )
28. number=checkforRemainingbits
29. select randomly metameric from shared
30. use its block to complete the message
31. change the code of the number of added bits in the block if needed
32. End

* By the enhanced pseudocode we aim to achive three goals:

1. Reduce The too big time spent by fixing code.
2. Avoid the infinite loop when executing fixing code
3. Make sure that there is no more than one metameric shares its block with an other block in the same chromosome, to negate possibility of message distoration.

* Mutation ratio was kept adaptive using the following formula:

mr = mr0\*exp(-genr/maxGen)

where mr is current mutation rate, mr0 is the initial mutation ratio, genr is the current iteration and maxGen is the number of iterations.

* Adding mutation ratio parameter to the mutation procedure for making more exploration like that:

Select mutationRatio\*length(solution) metameric then apply the mutation on them

* one problem has appeared in this approach, when the solution is mutated, then the mutation process may select the “bitBlanes” gene to mutate it, thus the number of bits that is hided with this metameric will be change to be smaller or higher, and then, the need for fixing code appeas.
* Mutated solution fixing pseudocode:

% Chromosome fixing pseudocode

Input**:**

Chromosome

trueMessgeBitsNumber

OutPut**:**

fixedChromosome

start**:**

counter **=** 1

subProcedure**:** count the fake bits number

**{**

**for** each chromosome metameric

bitplanesNumber **=** get the number of bits in this metameric bitplane

numberOfPixels **=** numberOfPixels in this metameric

bitsCount**(**counter**)** **=** numberOfPixels **\*** bitplanesNumber

counter **++**

**end**

fakeBitsNumber **=** sum**(**bitsCount**)**

**}**

**while** fakeBitsNumber **~=** trueMessgeBitsNumber

**if** fakeBitsNumber**>**trueMessgeBitsNumber

difference **=** fakeBitsNumber**-**trueMessgeBitsNumber

meta **=** select random metameric

pixelNumber **=** meta**.(**get pixels number**)**

bp **=** meta**.(**get the bitplanes**)**

numberOfBP **=** meta**.(**get the number bitplanes bits**)**

numberOfPixelsToDelete **=** ceil**(**difference**/**numberOfBP**)**

newPixelsNum **=** pixelNumber **-** numberOfPixelsToDelete

**if** newPixelsNum**<**1

newPixelsNum **=** 1

**end**

**else**

difference **=** trueMessgeBitsNumber**-**fakeBitsNumber

meta **=** select random metameric

pixelNumber **=** meta**.(**get pixels number**)**

bp **=** meta**.(**get the bitplanes**)**

numberOfBP **=** meta**.(**get the number bitplanes bits**)**

numberOfPixelsToAdd **=** ceil**(**difference**/**numberOfBP**)**

newPixelsNum **=** pixelNumber **+** numberOfPixelsToAdd

**end** **[**if\_fakeBitsNumber**>**trueMessgeBitsNumber**]**

fakeBitsNumber **=** call subProcedure

**if** fakeBitsNumber still greater than trueMessgeBitsNumber

**if** fakeBitsNumber **-** trueMessgeBitsNumber **==**1

**if** numberOfBP **==** 2

break\_the loop and ignore the last bitplane

**end**

**elseif** fakeBitsNumber **-** trueMessgeBitsNumber **<=**2

**if** numberOfBP **==** 3

break\_the loop and ignore the last bitplane

**end**

**elseif** fakeBitsNumber **-** trueMessgeBitsNumber **<=**3

**if** numberOfBP **==** 4

break\_the loop and ignore the last bitplane

**end**

**end**

**end** **[**if\_fakeBitsNumber still greater than trueMessgeBitsNumber**]**

**end** **while**

Project Workflow:

* When the proposed appraoch didn’t give us the satisful results, we resorted to change the population based algorithm to individual based algorithm, the reason for that is the hyper problem search space, that needs too big number of iterations number and population size. So, the individual based algorithm is more suitable for our problem with the hyper search space.
* 2 steps we’v done in this direction:

1. Using Tabu Search algorithm instead of the propsed genetic algorithm with keeping the same metameric solution representation.

* In TS algorithm, there is one solution that searches locally for the optimal value.
* Pseudocode

|  |
| --- |
| 1. Get the initial solution // initialized using the proposed metameric algorithm 2. For iter = 1 🡪 maxIter 3. newSol = mutationOperater(oldSol) // same operator that used in the proposed metameric algorithm 4. Evaluate(newSol) 5. If newSol was improved 6. oldSol = newSol 7. End 8. end |

1. merge the proposed ga with TS to get the benefit of the already designed crossover operator

|  |
| --- |
| // while the proposed algorithm runs:  For each 2 child solutions resulted after the crossover operator do:  For iter = 1🡪 number of TS iters // much smaller than maxIter   1. Child1\_mut = mutationOperator(child1) 2. If Child1 was improved   Child1 = Child1\_mut  Repeat 1 & 2 for child2 |

* pseudocode
* But the last developments also didn’t give us the satisful results compared with the benchMark algorithm.
* To approve the benefit of hiding the secret bits in a distributed areas of the image, we returned back to the GA benchMark algorithm that is already superior over all other algorithms, and make it decompose the secret bits to n part and then hide them in non-continous areas of the image.
* Decomposition Pseudocode:

Input:

imCover, secretMessage, n

Output:

stegno

Start:

1. subMsgs = decompose(secretMessage,n)
2. old\_hiding\_pixels = [ ]
3. finalChrom = [ ]
4. for subMsg in subMsgs
5. [bestChrom,stegno,hidingPixels] = benchMarkAlgo(imCover, subMsg)
6. old\_hiding\_pixels = [old\_hiding\_pixels hidingPixels]
7. imCover = stegno
8. finalChrom = [finalChrom bestChrom]
9. End for

End

* Modified cost function pseudocdoe

|  |
| --- |
| Input:  imCover,stegno,oldHidignPixels,hidingPixels  Output:  Cost  Start:   1. If any(ismember(hidingPixels, oldHidignPixels)) 2. Cost = inf 3. Else 4. Cost = MSE(imCover,stegno) 5. End if   End |

By implementing the above approach, we solve the problem of hyper search space that appears in the metameric approach, and the continuity problem that appears in the benchMark approach.

Execution:

Execution parameters:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Population size | maxIter | number of TS iters | Mutation rate/mutation ratio |
| GA-BenchMark | 50 | 50 | - | Gaussian mutation with default matlab ga values for scale and shrink that are modified adaptivally using the generation number , default initial scale/shrink = 1/1 |
| GA-Metameric | 50 | 50 | - | 0.2/0.5 |
| TS-Metameric | 1 | 50\*50 | - | -/05 |
| GATS-Metameric | 50 | 50 | 5 | -/0.5 |
| GA-BenchMark-2-decomposed | 50 | 50 | - | Gaussian mutation with default matlab ga values for scale and shrink that are modified adaptivally using the generation number , default initial scale/shrink = 1/1 |
| GA-BenchMark-3-decomposed | 50 | 50 | - | Gaussian mutation with default matlab ga values for scale and shrink that are modified adaptivally using the generation number , default initial scale/shrink = 1/1 |

Other uniform params:

1. image dims scaling = [512,512]
2. Fitness function = -MSE
3. Secret Message: Steganography is the practice of concealing a message within another message or a physical object. In computing/electronic contexts, a computer file, message, image, or video is concealed within another file, message, image, or video. The word steganography comes from Greek steganographia, which combines the words steganos , meaning "covered or concealed, and -graphia  meaning "writing.The first recorded use of the term was in 1499 by Johannes Trithemius in his Steganographia, a treatise on cryptography and steganography, disguised as a book on magic. Generally, the hidden messages appear to be (or to be part of) something else: images, articles, shopping lists, or some other cover text. For example, the hidden message may be in invisible ink between the visible lines of a private letter. Some implementations of steganography that lack a shared secret are forms of security through obscurity, and key-dependent steganographic schemes adhere to Kerckhoffss principle.The advantage of steganography over cryptography alone is that the intended secret message does not attract attention to itself as an object of scrutiny. Plainly visible encrypted messages, no matter how unbreakable they are, arouse interest and may in themselves be incriminating in countries in which encryption

Metameric algorithms specific parameters:

QL = 100; % quantization level

T\_nbc = 400; % number of bit change threshold

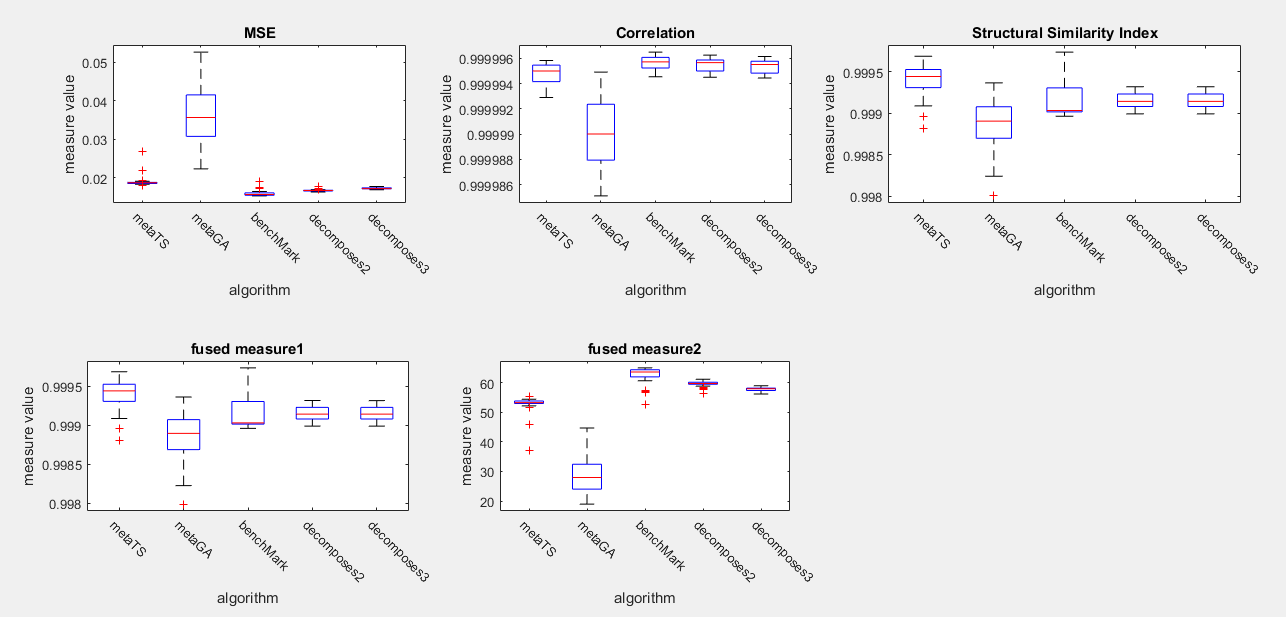
BenchMark Decomposed specific parameters:

We have executed the previous approach using n=2 and n=3, this means that the message was decomposed to 2 and 3 subMsgs respectively

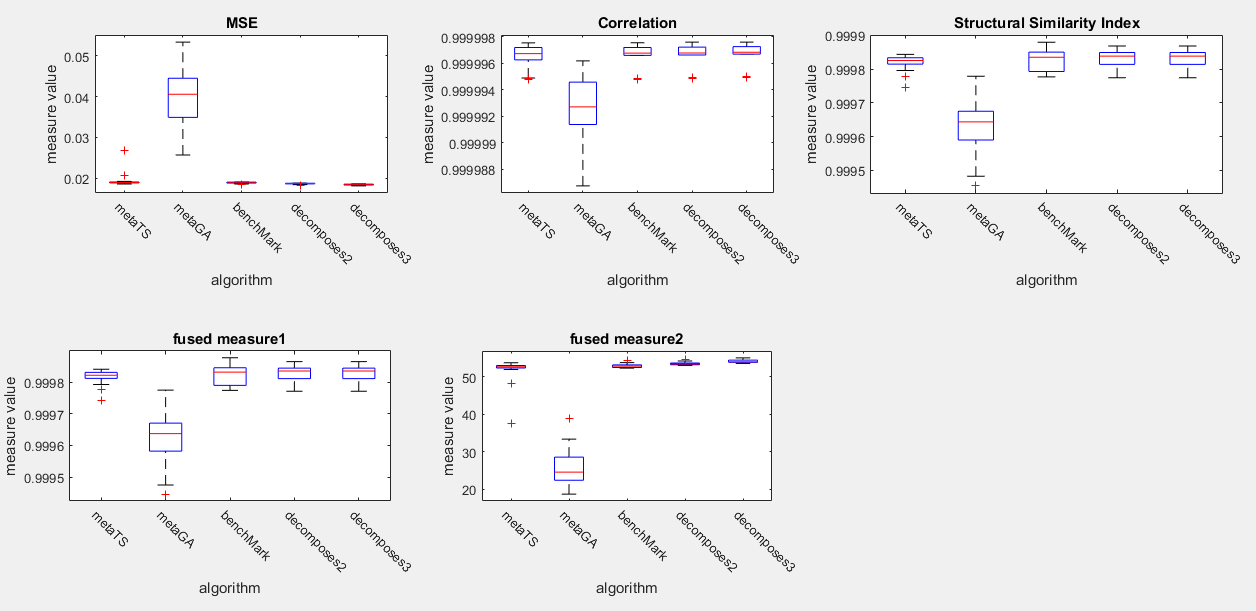
Used Measures:

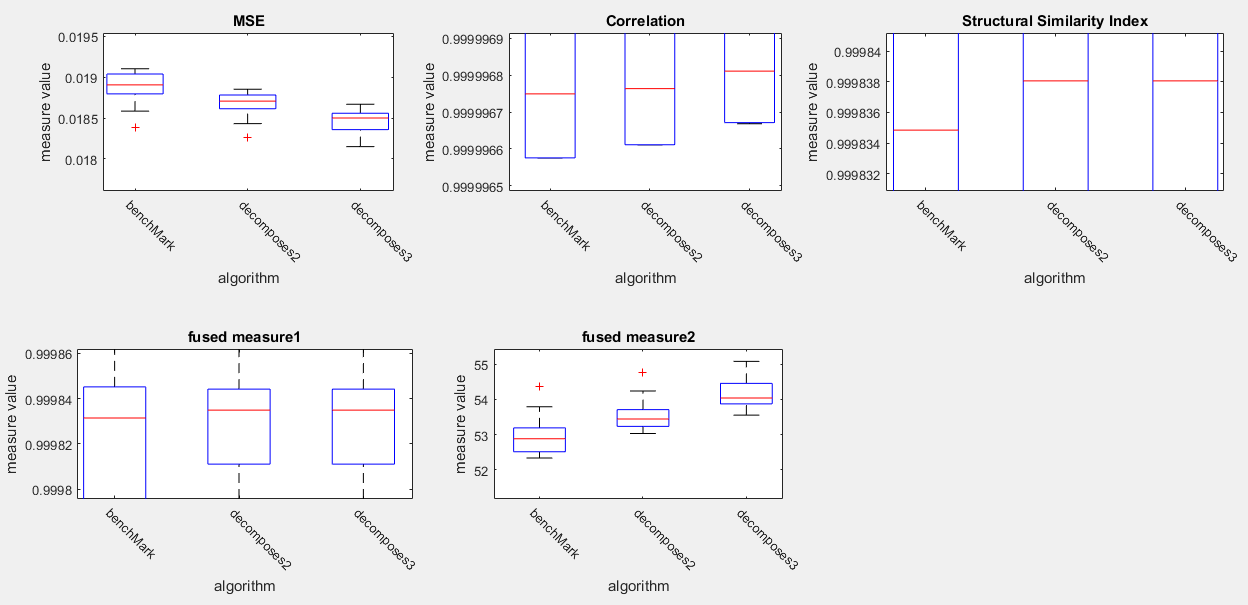
1. MSE: mean of squared errors
2. Correlation
3. Structural similiarity
4. Fused Measure1 = correlation \* structural similiarity
5. Fused Measure2 = Fused Measure1 / MSE

Results:

1. Brain Dataset within 5 images and 5 seeds for each image (those results excepts GATS algorithm because of its long execution time)

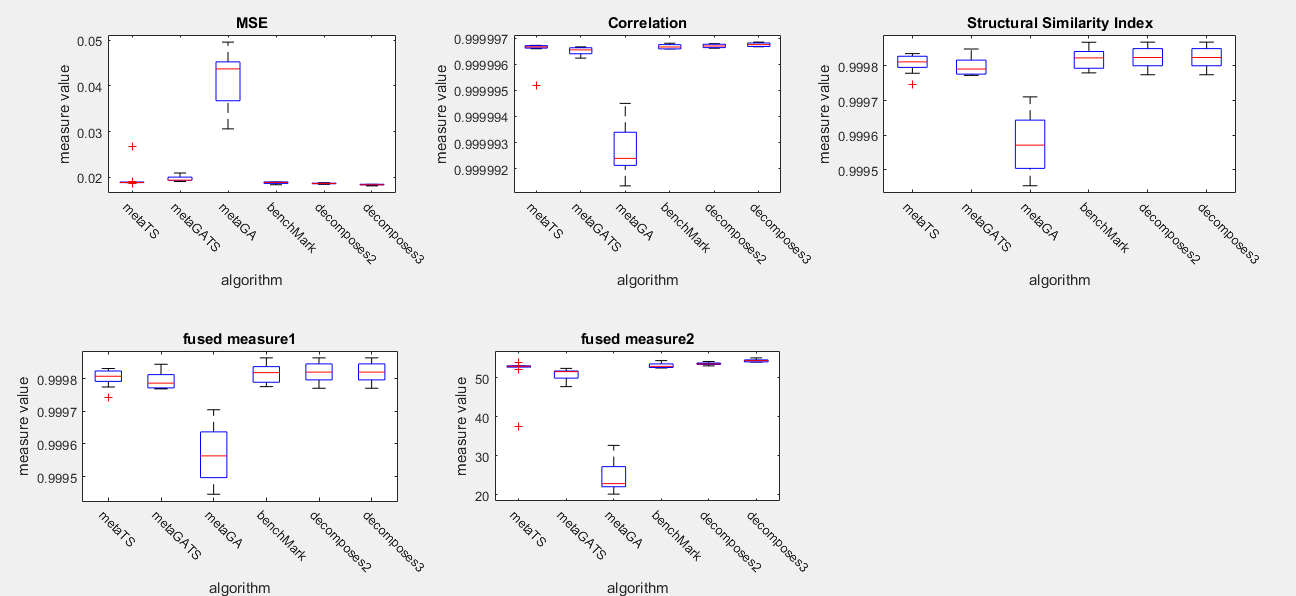
We notice that the benchMark was the best in terms of MSE in this data, but Decomposed to 2 and 3 are superior in terms of structural similiarity. a possible factor that may affect the MSE of decomposed stegno that the benchMark embed the chromosom in the image, while we don’t embed the chromosome in our last developed metameric approach, and since we finally return to the benchMark and decompose the message to n sub message, there become n metameric, and those metamrics need to be embedded in the image as the benchMark algorithm does, this leads to increase the number of distored pixels, as well as the MSE, althouh we have used small values for n, but the difference between the decomposed approach and the benchMark is also small.

1. Chest Dataset : within 5 images and 5 seeds for each image (those results excepts GATS algorithm because of its long execution time)

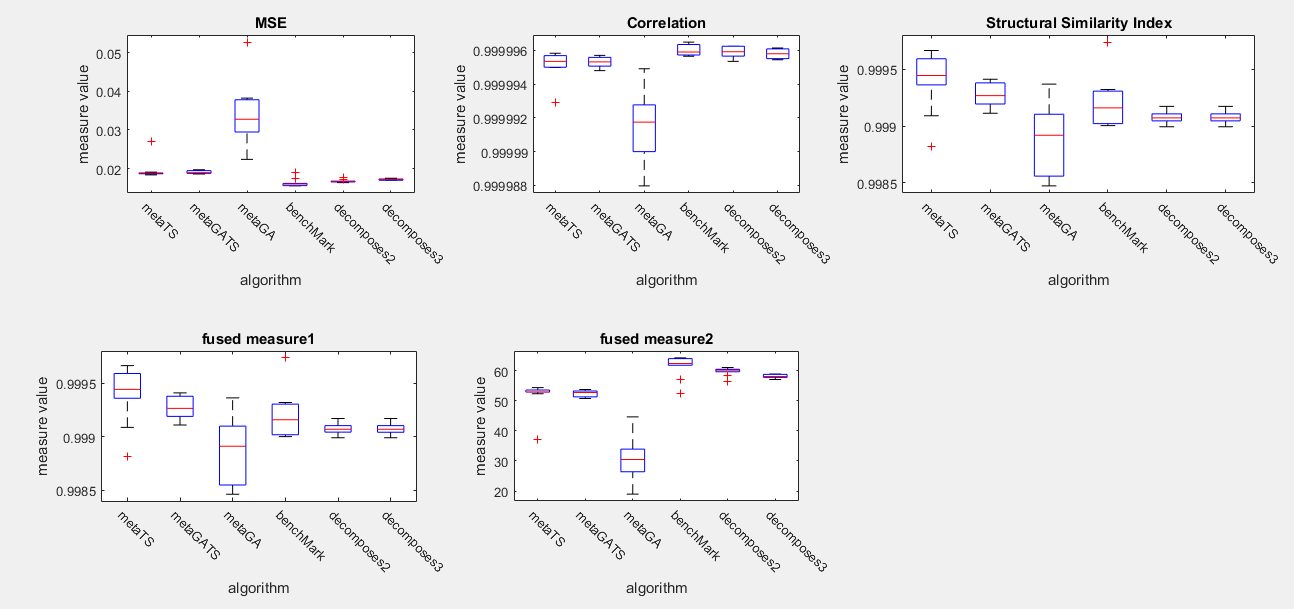
By zooming in

We notice that the decomposed to 2 and 3 are superior in terms of all measures in the Chest data, I guess that decomposing of message is more effective in the more hetrogenious images like chest images.

- The results with metaGA\_TS algorithm for chest data but only 2 images and 5 seeds for each image for Chest Dataset



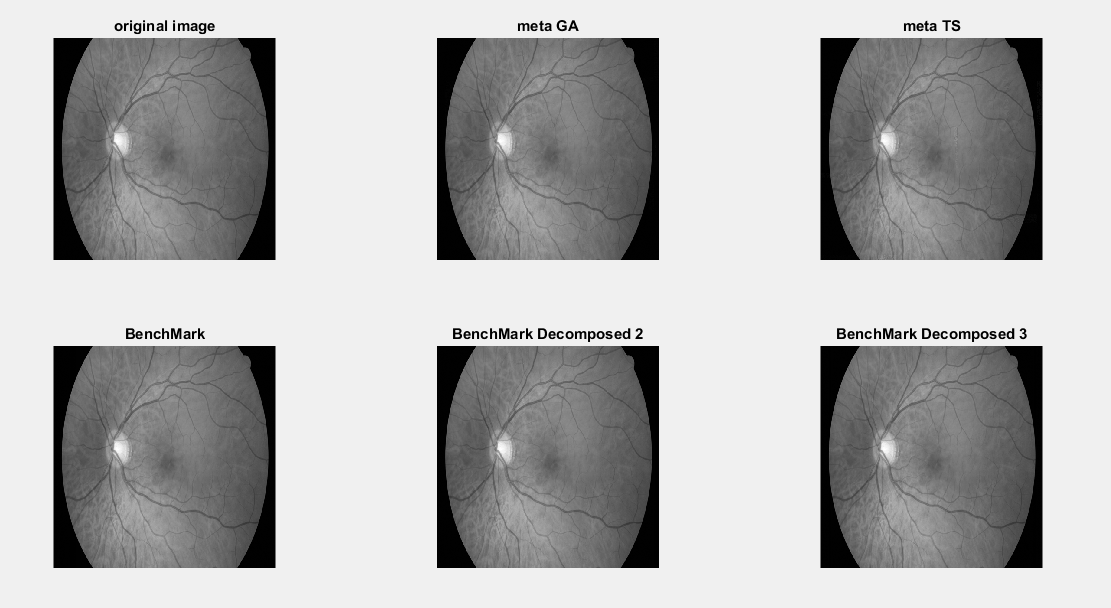
- The results with metaGA\_TS algorithm for chest data but only 2 images and 5 seeds for each image for Brain Dataset



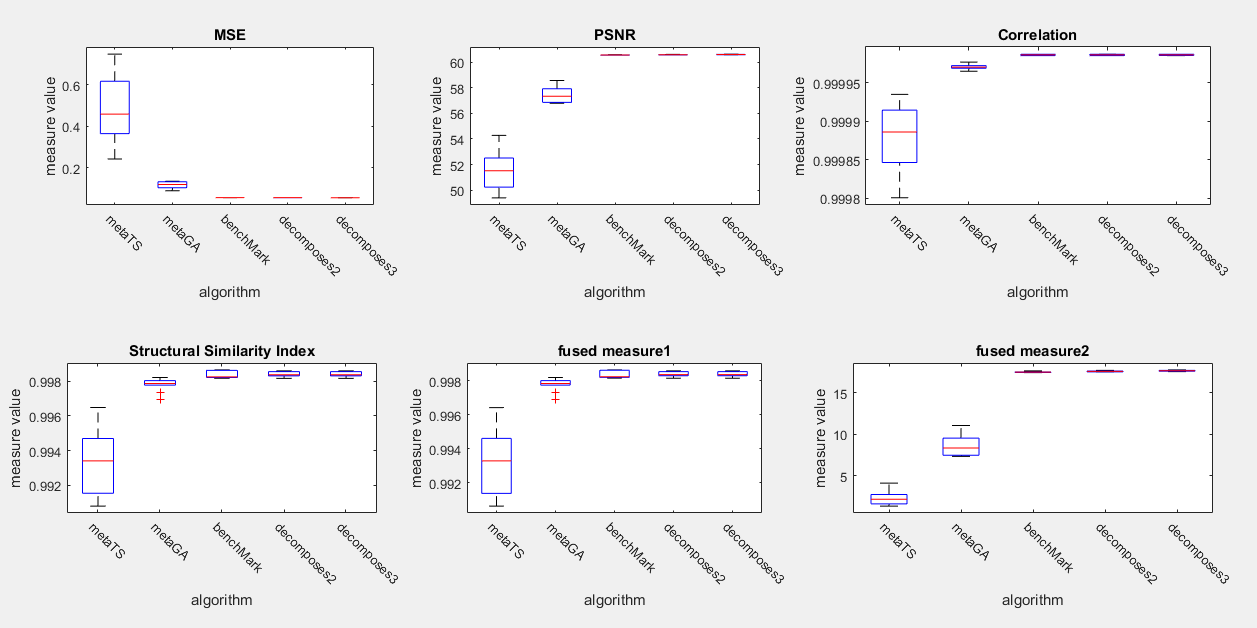
1. Rentina Dataset:

After we found that the decomposed approaches performed better in the less homogenuse images, we used the Rentina dataset because its images are not homogenuse, and the decomposed approaches have performed the best on that dataset

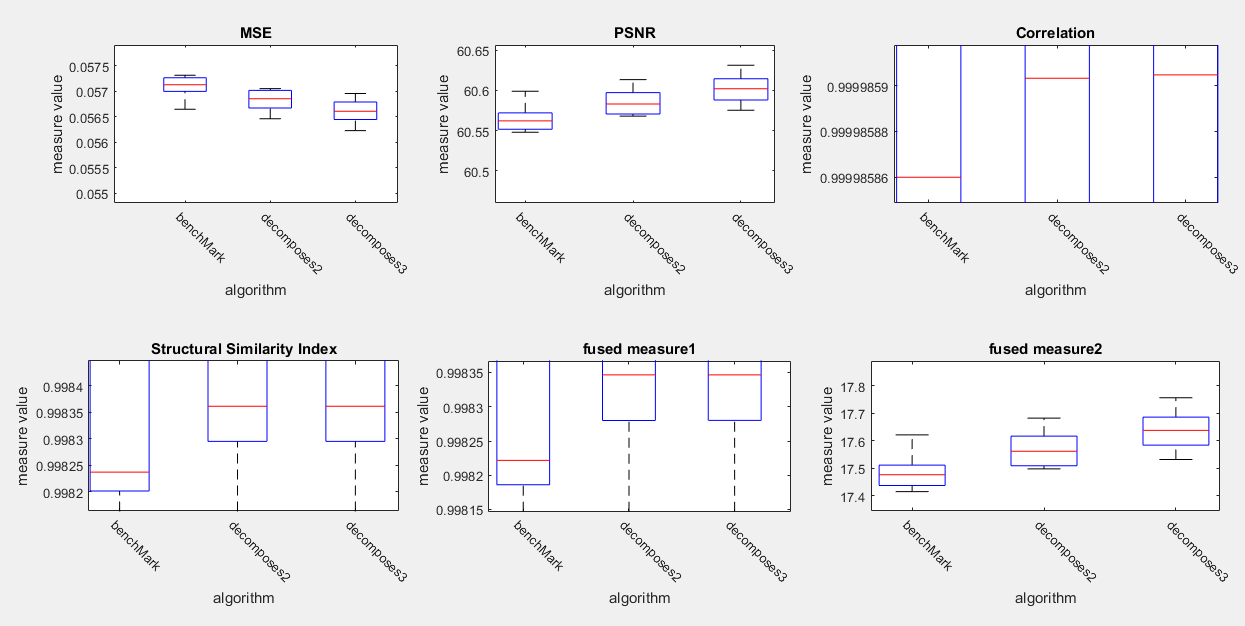
Example of an image and the different algorithms stegno images



The results:

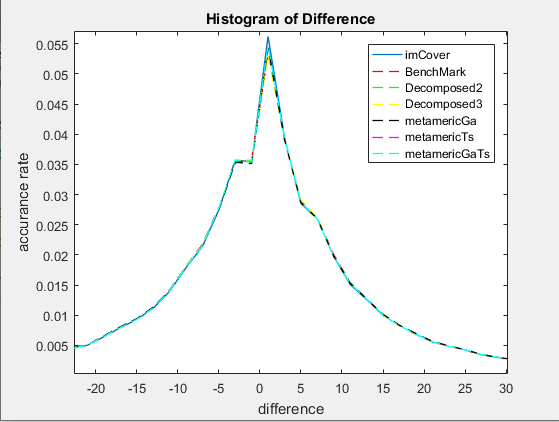


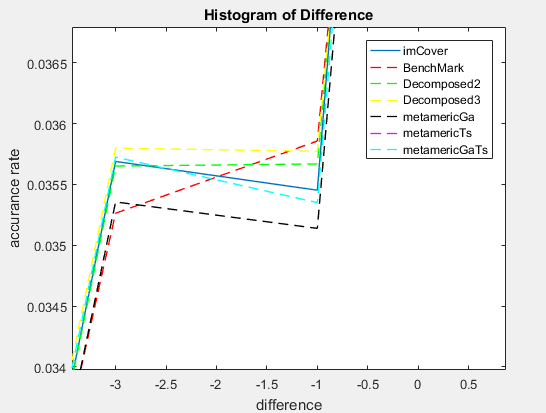
Zooming in:



So, we notice that the decomposed algorithms are the best when the images contain more changes in the pixels values.

Security Measures Analysis:

1. Histogram of difference example:

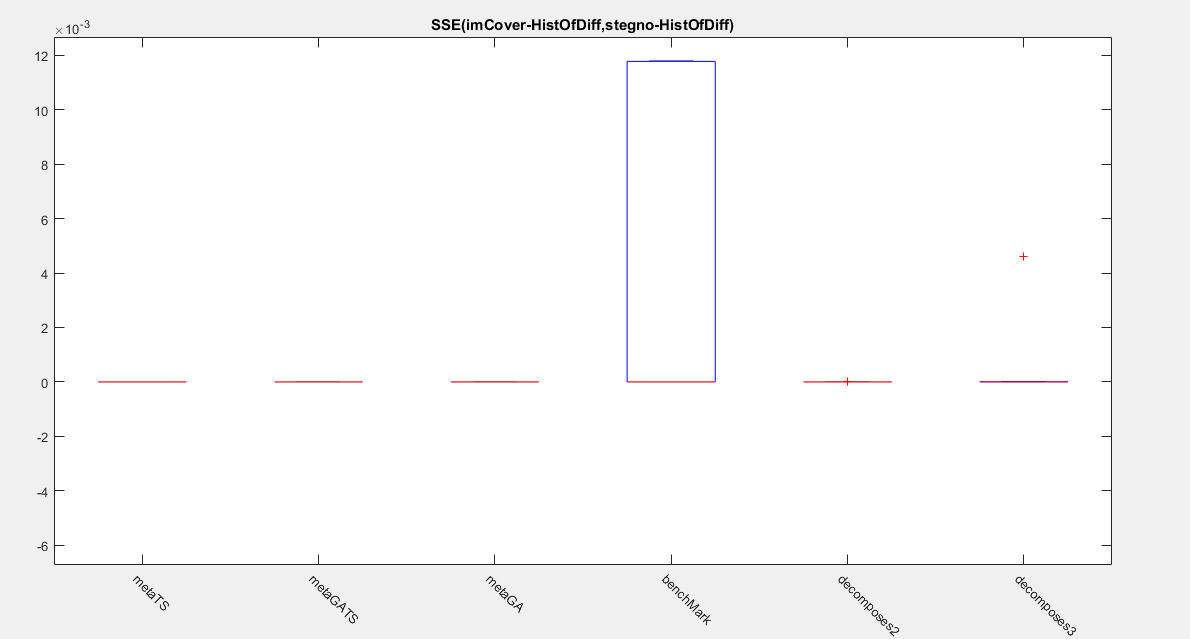
Zooming in:

So, it is clear that there is no significant difference between the different approaches.

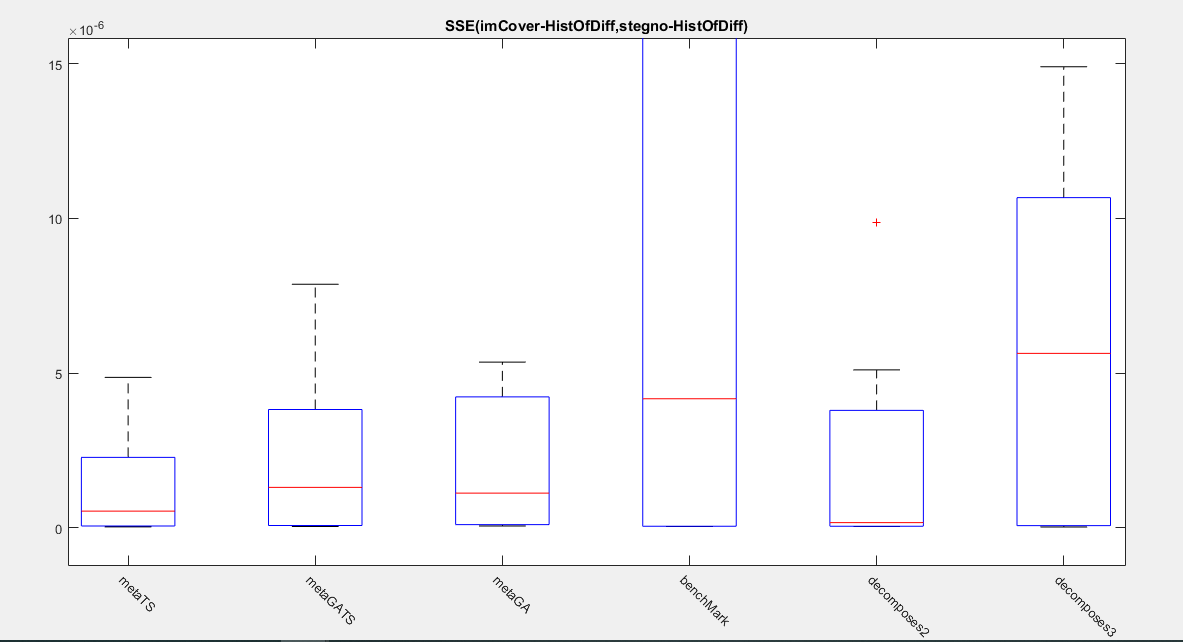
* To illustrate wich algorithm is the best in terms of Histogram of difference, we compute the sum of squared errors between the histogram of the image cover and the others.

SSE = Ʃ

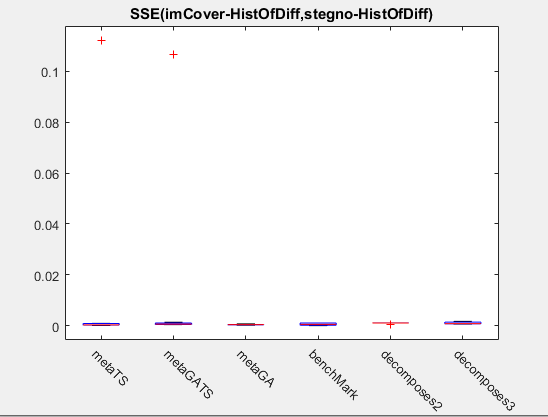
* SSE Results for Chest Dataset:

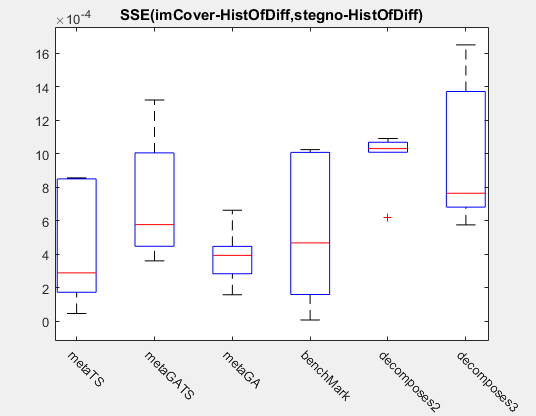


* Zooming in:



* SSE Results for Brain Dataset:

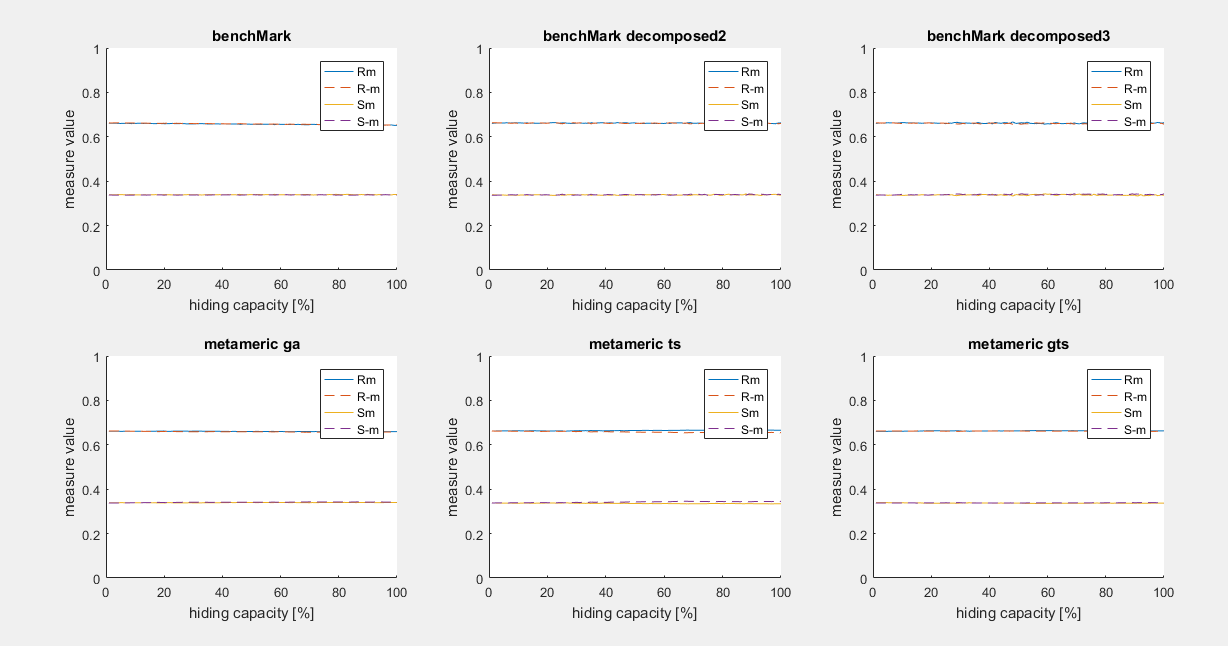


* Zooming in:
* 

We notice that the metameric algorithms are more secure in terms of Histogram of difference

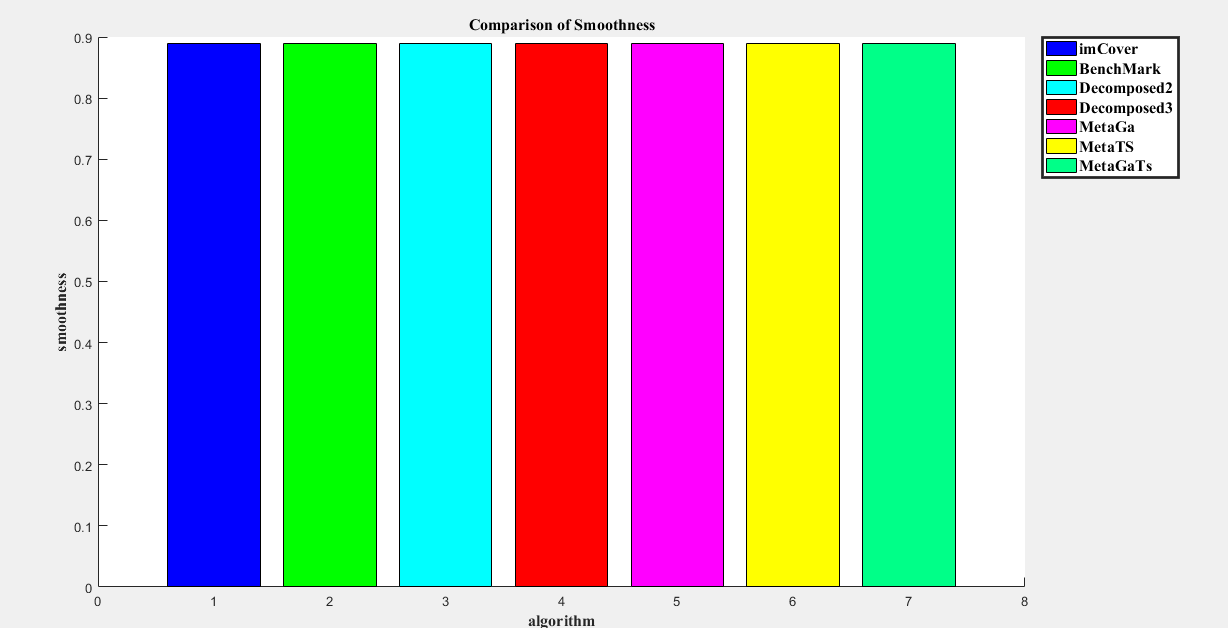
1. RS analysis example:

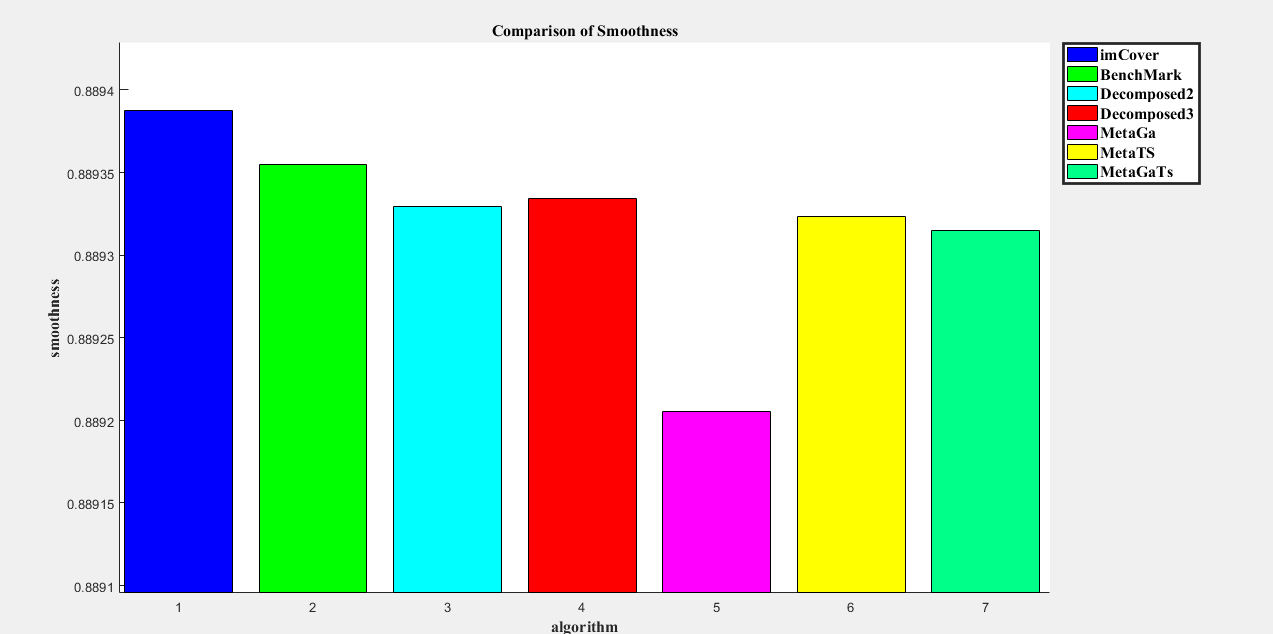
We took example of the best chromosom for random image and embidded the message gradually in the image cover with 100 steps. And used maske with size of 32 pixels



As we notice, there is also no significant difference between the different algorithms in terms of RS measure

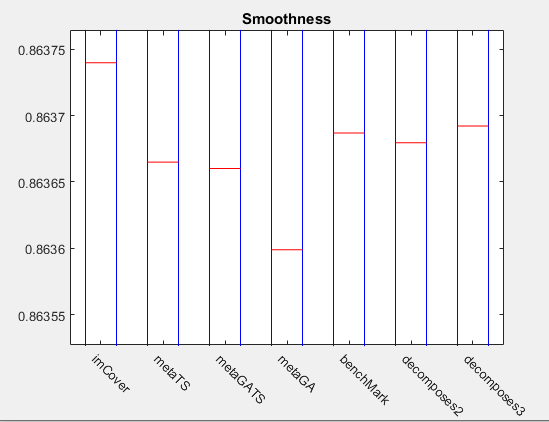
1. Smoothness Measure:



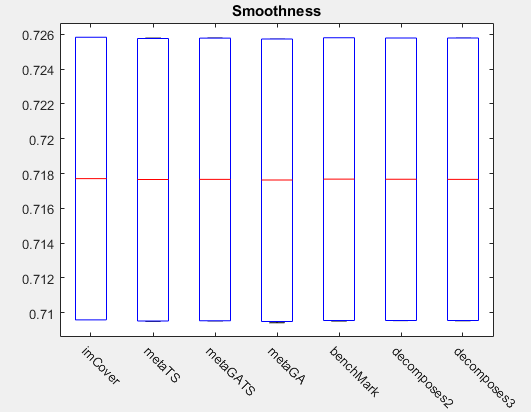
Zooming in:

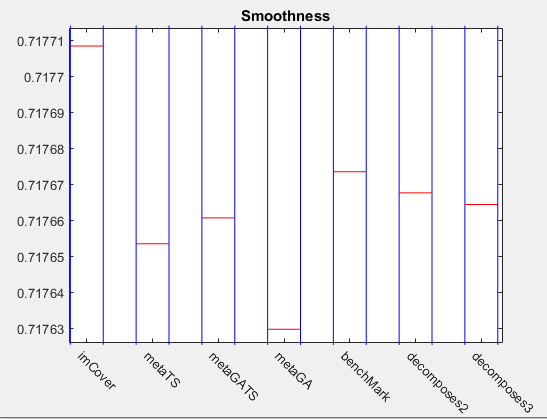
In this example, the benchMark approach is the best in terms of smoothness.

* Statistical results for smoothness measure:
* 1- Chest Dataset

Zooming in:

* 2- Brain Dataset:



* Zooming in:
* We find tha the benchMark is the best in terms of smoothness.
* The most likely reason in my opinion is that the smoothness calculation is based on the gradient of the image, so the gradient will be more susceptible to changes when we distribute the secret bits in many blocks.