Отчет по эксперименту с модифицированным алгоритмом PM1

Вспомогательные функции

Для реализации работы данного алгоритма был создан ряд дополнительных функций:

1. cut_image_into_blocks(im, n=8) – на вход подается одноканальная картинка в формате np.array и опционально размерность блоков, на которые нужно разделить изображение (по умолчанию равна 8). На выходе возвращает 4-мерный массив, первые две координаты которого являются координатами блока изображения, а последние два – координатами пикселя в выбранном блоке.
2. concatenate_image(im) – функция, обратная предыдущей, на вход получает четырехмерный массив, поделенный на блоки, на выходе возвращает np.array, представляющий из себя матрицу пикселей исходного изображения, созданную путем склеивания блоков входного массива.
3. count_v_i(block, n=8), count_w_i(block, n=8) – функции, необходимые для подсчета коэффициентов V и W, описанных в статье для пересортировки используемых блоков DCT коэффициентов.
4. g_j(j) – функция, которая возвращает вес коэффициента в зависимости от его порядкового номера (для более высоких частот больший вес).
5. dct2d(a) – функция, возвращающая DCT-коэффициенты входной матрицы a.
6. idct2d(a) – функция, применяющая обратное DCT-преобразование ко входной матрице и возвращающая результат этого преобразования.
7. get_dct_coefs(im) – функция, которая принимает на вход четырехмерный массив, применяет DCT-преобразование к каждому блоку и возвращает результат в виде четырехмерного массива DCT-коэффициентов.
8. get_image_from_dct_coefs(im) – функция, которая принимает на вход четырехмерный массив, применяет обратное DCT-преобразование к каждому блоку и возвращает результат в виде четырехмерного массива пикселей полученного изображения.
9. insert_message_PM1(image, message) – функция, реализующая алгоритм встраивания сообщения в изображение, описанный в статье. В целом функция полностью идентична описанной в статье, за исключением того, что мною была добавлена обработка исключительной ситуации, когда f становится отрицательным, что означает, что в изображении закончились ненулевые коэффициенты. Функция возвращает стегопуть U и стегоконтейнер содержащий сообщение message.
10. extract_message(im, route) – на вход принимает стегоконтейнер и стегопуть, сформированный в предыдущей функции, возвращает извлеченное из контейнера сообщение.
11. PSNR(original, compressed) – функция, возвращающая значение PSNR между двумя входными изображениями.
12. count_BER(a, b) – функция, возвращающая значение BER между двумя входными списками a и b.
13. get_metrics(im, mes) – функция, принимающая на вход изображение и сообщение, которое необходимо в него встроить, возвращается значение метрик psnr, ssim и ber в указанном порядке.
14. custom_plot(x, y, ax=None, **plt_kwargs) – функция, которая строит график зависимости y от x либо на заданном объекте ax (matplotlib.pyplot.Axes), либо создает такой объект сама и строит график на нем.

Проведение эксперимента

Часть 1

Для проведения эксперимента была написана функция make_experiment, принимающая на вход список файлов-изображений, которые будут использованы в качестве стегоконтейнеров. В функции заданы два важных параметра – MAX_VALUE, обозначающий максимальную длину встраиваемого сообщения, и PERCENTAGES – список долей, обозначающих заполняемость контейнера на каждом шаге эксперимента. В данном эксперименте использовались следующие доли – 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. MAX_VALUE = 153843, это количество ненулевых ДКП-коэффициентов у изображения с наименьшим количеством таких коэффициентов («motion06.512.tiff») среди всех выбранных изображений. Далее для каждого числа p, содержащегося в PERCENTAGES проводится эксперимент:

1. Вычисляется параметр m_len = round(p * MAX_VALUE), обозначающий длину встраиваемого сообщения.
2. Генерируется сообщение m
3. Создаются пустые массивы p, s, b и t, которые будут хранить значения метрик на данном шаге.
4. Далее в каждую картинку из входного списка встраивается и извлекается сообщение, при этом происходит подсчет необходимых метрик.
5. После обработки всех картинок средние значения метрик для данного объема заполняемости заносятся в массивы (PSNR_VALUES, SSIM_VALUES, BER_VALUES, TIME_VALUES), хранящие соответствующие метрики.
6. Как только заканчивается проведение эксперимента для всех указанных объемов, функция возвращает массивы с метриками в следующем порядке (PSNR_VALUES, SSIM_VALUES, BER_VALUES, TIME_VALUES).

Часть 2

Далее в рамках эксперимента тестировался подход с итеративным исправлением ошибок, который должен уменьшить количество возникающих ошибок при извлечении сообщения из стегоконтейнера. Для проверки этого метода было написано несколько новых функций:

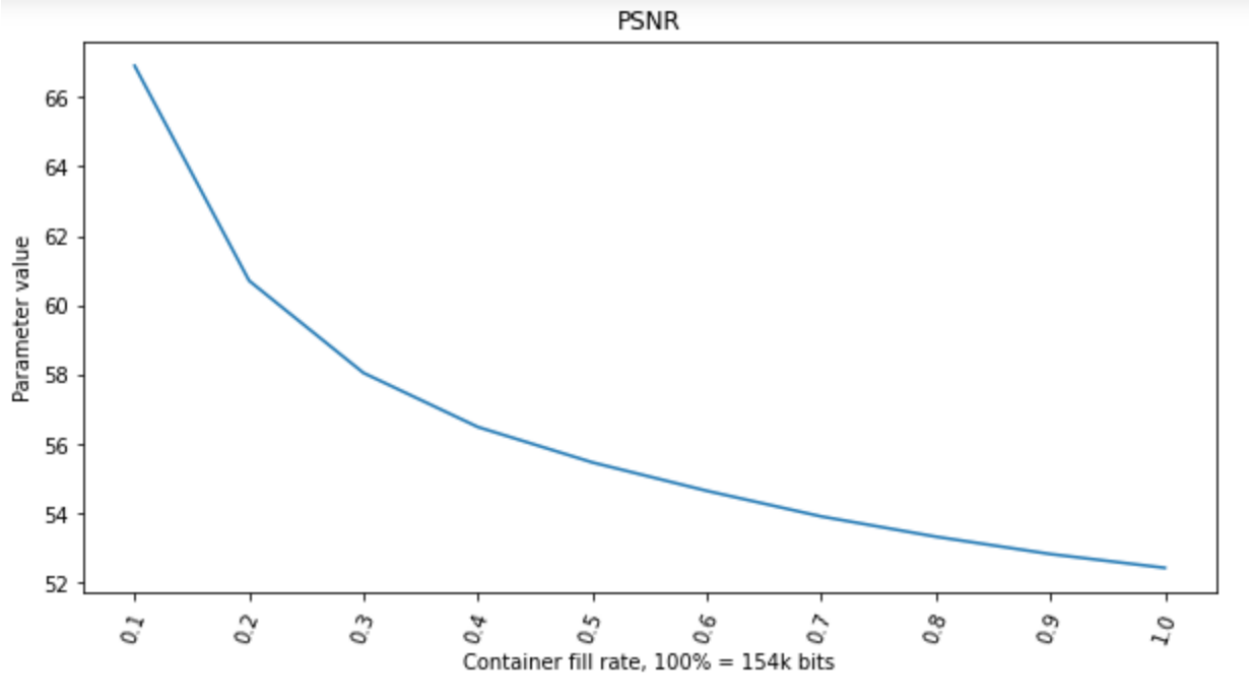
1. get_block_indexes_structure_from_route(route)– принимает на вход путь, сформированный при вставке сообщения в методе insert_message_PM1. Возвращает словарь из 3 компонентов – block_order (список, показывающий порядок обхода блоков), coefs (словарь, который для номера блока возвращает список коэффициентов, которые нужно модифицировать) и indexes (словарь, который для каждого блока возвращает позиции битов в исходном сообщении, которые должны быть спрятаны в этом блоке).
2. get_mes_from_block(block, coefs)– на вход поступает блок изображения и порядковые номера коэффициентов, в которых спрятаны необходимые биты и возвращает сообщение, которое спрятано в указанных коэффициентах данного блока.
3. embed_mes_into_block(block, mes, coefs) – принимает на вход блок изображения, сообщение и список коэффициентов, в которых необходимо спрятать биты сообщения. Возвращает блок изображения со спрятанным сообщением.
4. get_metrics_iterative_embedding(im, mes, threshold=10) – принимает на вход сообщение, изображение и пороговое количество итерация встраивания. В самом начале функция выполняет обычное встраивание и получает стегоконтейнер и стегопуть. Далее, в случае когда BER != 0, она производит итеративное встраивание в порядке, указанным в стегопути. Отличие от алгоритма, описанного в статье,

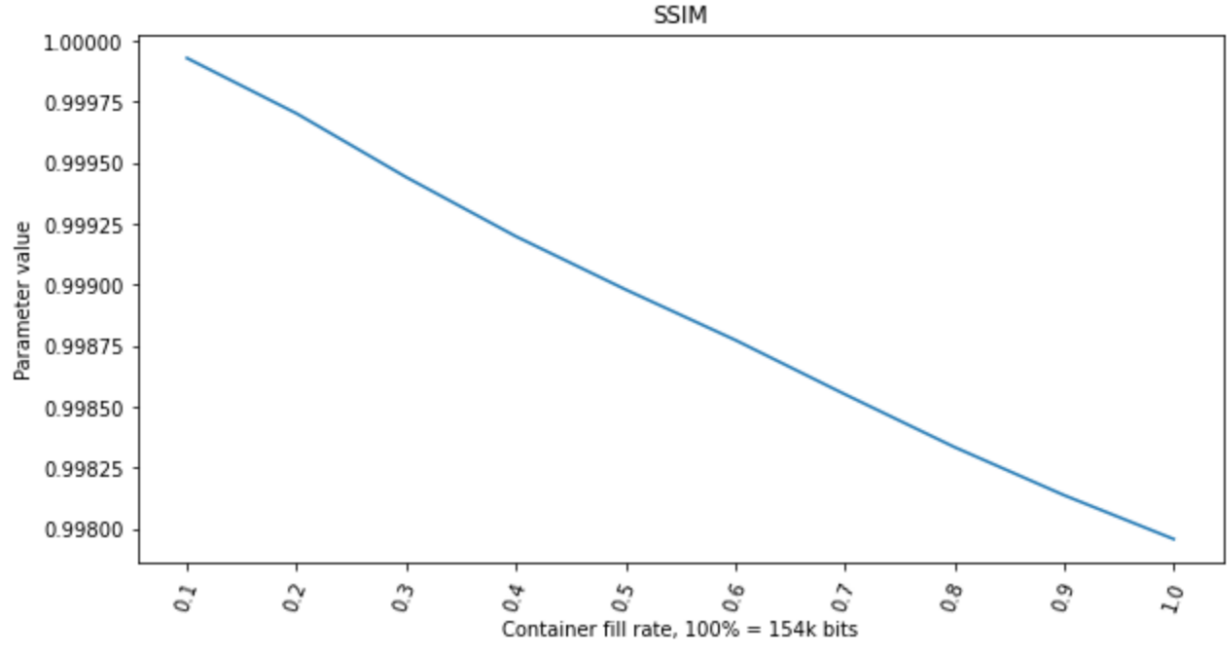
заключается в том, что при выполнении запоминаются все полученные варианты встраивания и затем выбирается лучший из них. Сделано это было потому, что в некоторых случаях процедура итеративного встраивания ухудшала имеющийся результат, поэтому необходимо было выбрать лучший вариант из полученных. Функция возвращает метрики PSNR, SSIM и BER

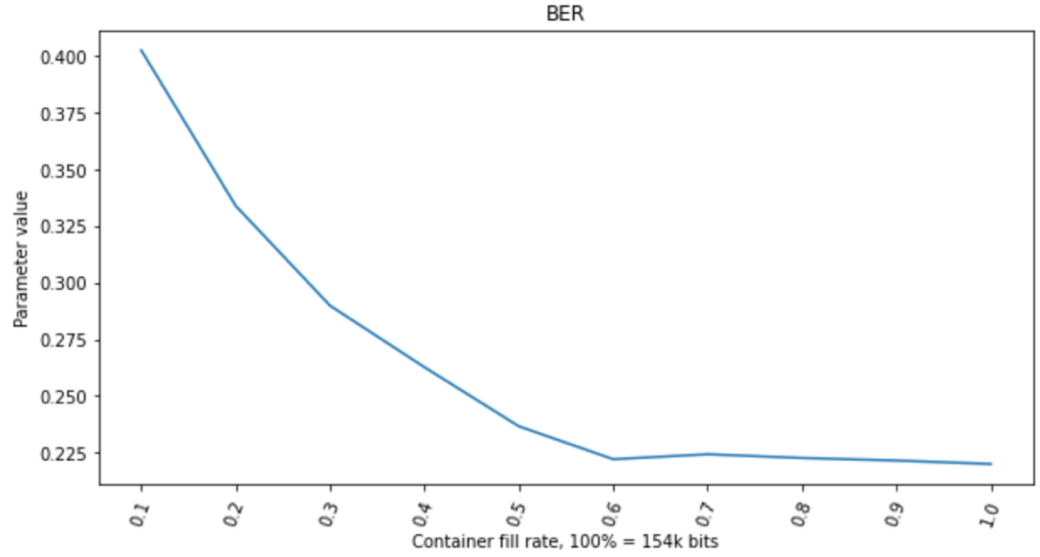
Результаты эксперимента

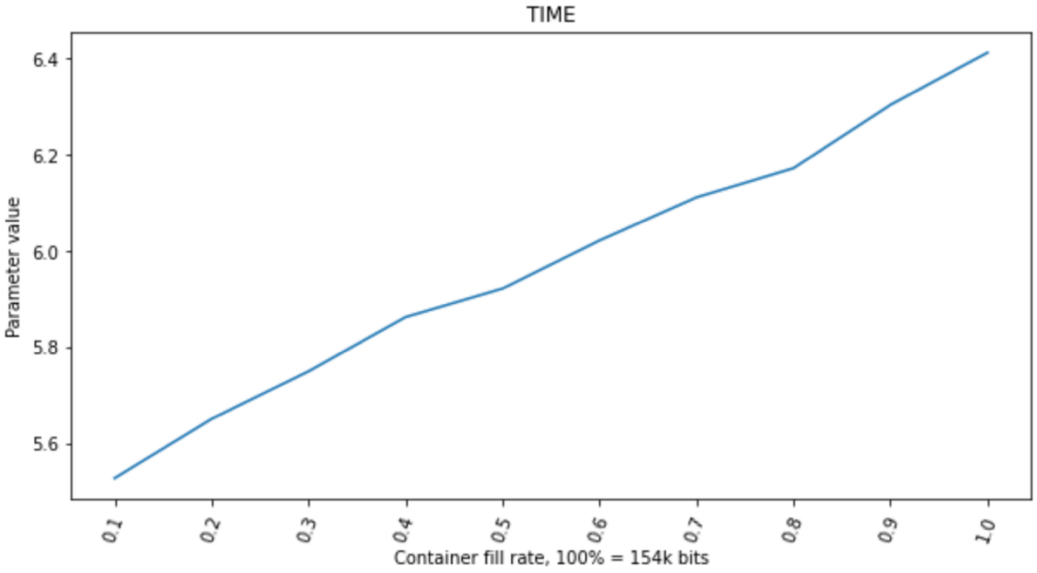
Как уже было сказано, в результате опытов было получено минимальное количество ненулевых коэффициентов ДКП для изображений – 153843. Именно столько их у изображения “motion06.512.tiff”, и это число мы возьмем за 100% заполняемость контейнера. В рамках эксперимента сообщения разной длины встраивались в различные черно-белые изображения, в качестве результатов бралось среднее значение метрик по всем изображениям. Вот график зависимости метрик и заполняемости контейнеров.

Часть 1 – значения метрик для PM1

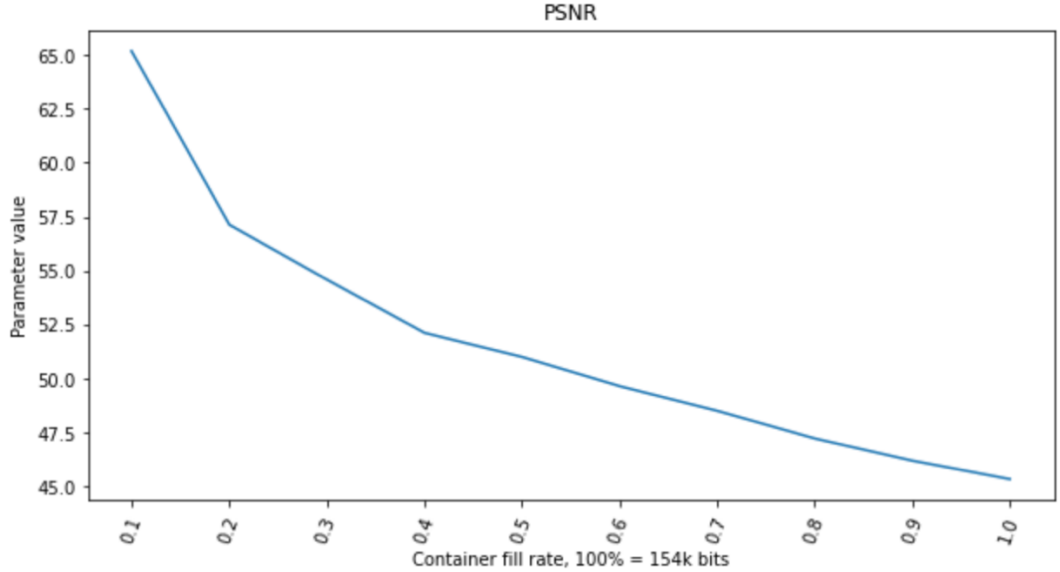


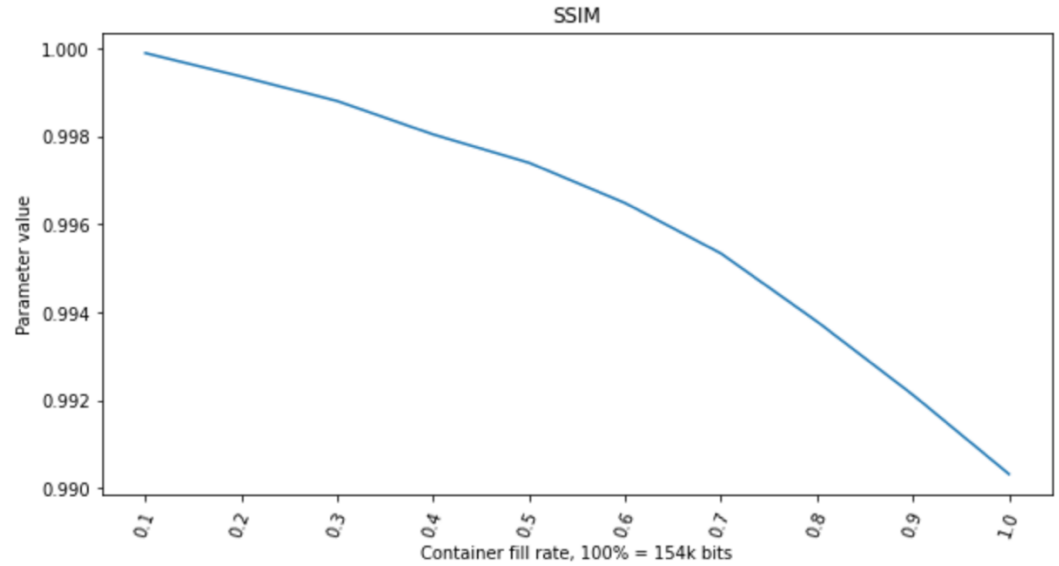


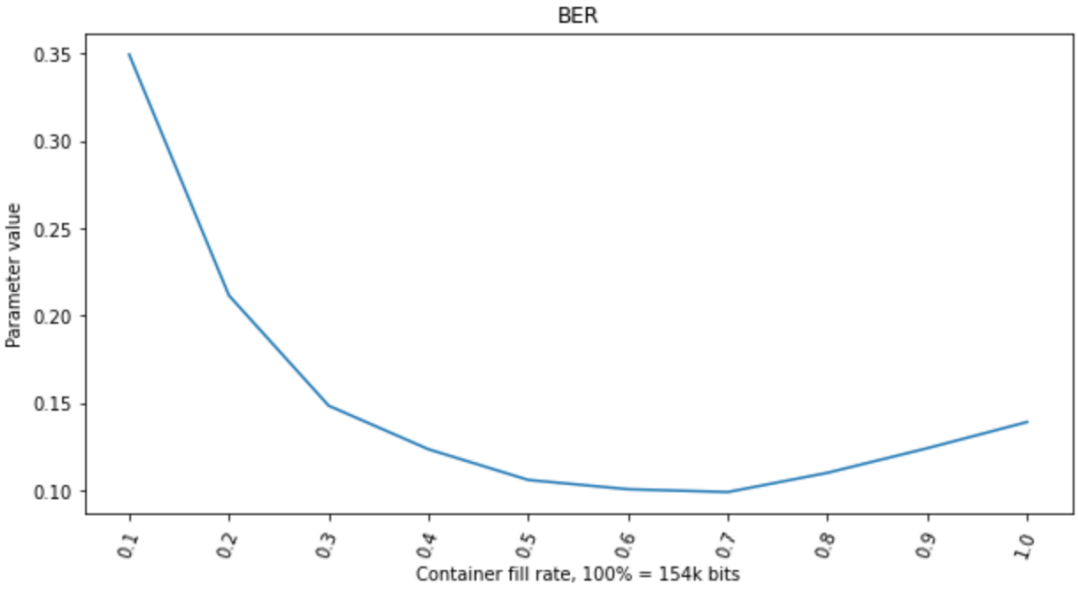




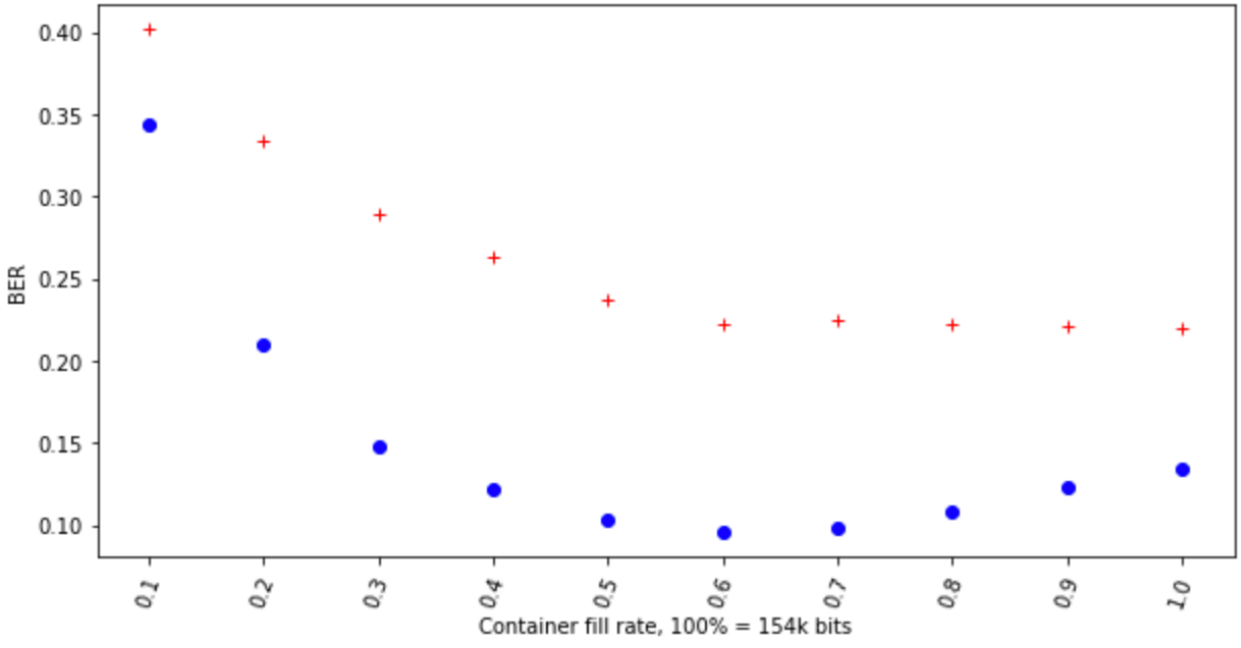
Часть 2 – значения метрик для PM1 с итеративным встраиванием







Часть 3 – Наглядное сравнение значений BER для двух методов



На данном графике синими точками отмечены значения BER для метода с итеративным встраиванием, а красными плюсами – значения BER для обычного встраивания.

Выводы

В результате эксперимента было выявлено улучшение качества встраивания итеративным методом по сравнению с обычным. В среднем показатель BER уменьшался на 10%. В рамках эксперимента также было выявлено, что при малом объеме встраиваемой информации BER принимает высокие значения, что может быть обусловлено особенностями работы алгоритма, который в приоритете встраивает информацию в высокочастотные коэффициенты. У итеративного метода имеются и недостатки – из-за частого применения повторного встраивания исходное изображение сильнее искажается, что приводит к уменьшению PSNR и SSIM, а также увеличению времени работы алгоритма (в среднем в 2 раза).

Ссылки

Все исходные коды и изображения находятся в репозитории:
https://github.com/Noktyrn/DigitalSteganogarphy