Национальный Исследовательский Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №7

по дисциплине «Цифровая обработка сигналов»

Выполнил:Ларочкин Г.И

Группа: P3400

Преподаватель: Тропченко А.А.

Санкт-Петербург

2020 г.

## Постановка задачи

**Задание №1.**

1. Используя соответствующие функции пакета Matlab, напишите программу, реализующую алгоритм сокрытия данных в младшем разрядном срезе изображения.
2. Проведите эксперименты по сокрытию данных и их извлечению для различных изображений и типов скрываемых данных (бинарных изображений, данных типа .txt, .dat).
3. Определите количественные характеристики, определяющие соответствие между пустым и заполненным стегоконтйнером.

**Задание №2**

1. Самостоятельно реализуйте алгоритм сокрытия данных (текста) в псевдо-белых и псевдо-черных пикселах изображения и с использованием средств пакета Matlab напишите программу, реализующую данный метод.
2. Выполните п.п. 2 и 3 задания №1.
3. Сравните емкость стегоконтейнеров для обоих методов.
4. Самостоятельно провести подобные манипуляции с несколькими различными изображениями. Привести вид исходного стегоконтейнера и гистограмму яркости пикселов подлежащего сокрытию бинарного изображения, стегоконтейнера после помещения в него скраваемой информации (с гистограммой) и вид восстановленного бинарного изображения (для каждого из изображений). Проанализировать недостатки и достоинства методов и вариант модификации для помещения скрываемой текстовой (символьной) информации.

# Решение задачи

## Реализация сокрытия данных в младшем разрядном срезе

В качестве шифруемых данных я использовал вотермарку:



Рис. 1 – вотермарка 64x64x1



Рис. 2 – исходное изображение 512x512x3

В данном алгоритме шифруется 1 бит необходимых данных в каждом пикселе исходного изображения. Таким образом, вместимость стего-контейнера для рисунка 2 составляет 98304 байта. Однако из-за этого некоторые пиксели теряют яркость, но это не заметно человеческому глазу. Коэффициент корреляции составил 0.9999.

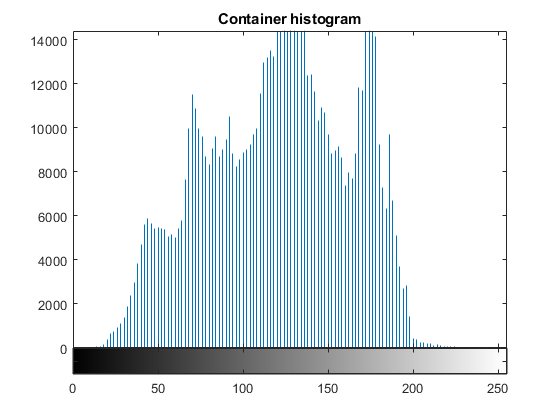
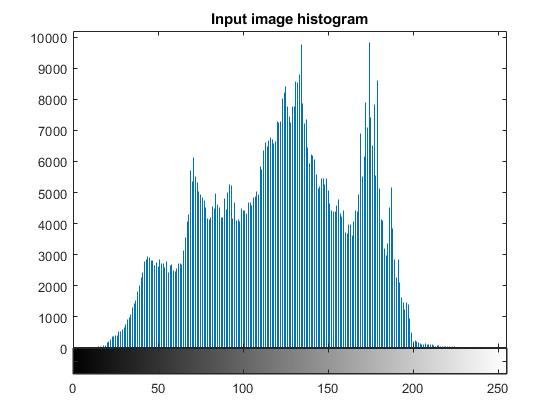


Рис. 3 – гистограмма исходного изображения и гистограмма контейнера

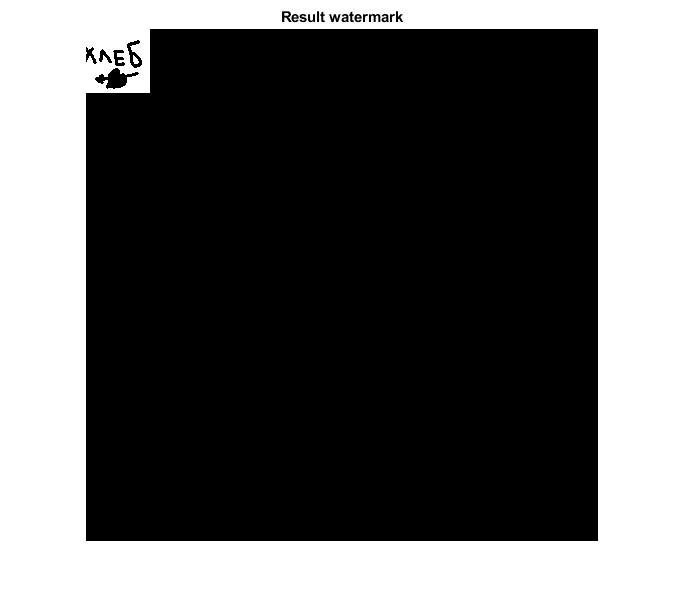
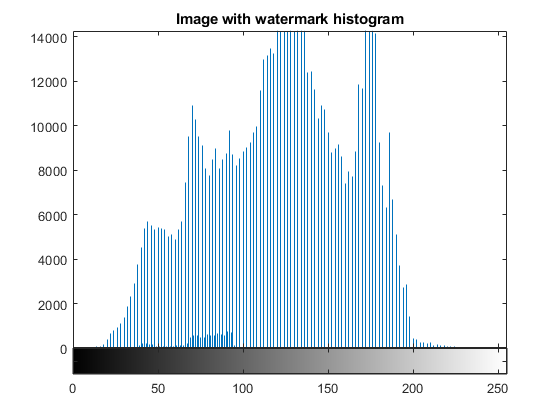


Рис. 4 – гистограмма изображения с вотермаркой и полученной из изображения вотермарки

Part1.m:

I = imread('resources/gray.bmp');

%I = imresize(I, 15); % to encrease the number of black/white pixels

W = imread('resources/watermark.bmp'); % watermark

[C, capacity] = getContainer(I);

fprintf("Container capacity: %0.1f bytes\n", capacity);

IW = C; % image with watermark

for i = 1:size(W, 1)

for j = 1:size(W, 2)

for k = 1:size(IW, 3)

if (W(i, j) > 0)

IW(i,j,k) = IW(i,j,k) + 1;

end

end

end

end

figure, imhist(I)

title('Input image histogram')

figure, imhist(C)

title('Container histogram')

figure, imhist(IW)

title('Image with watermark histogram')

W = IW;

for i = 1:size(IW, 1)

for j = 1:size(IW, 2)

for k = 1:size(IW, 3)

IW(i,j,k) = bitand(IW(i,j,k), uint8(254));

W(i,j,k) = (W(i,j,k) - IW(i,j,k)) \* 256;

end

end

end

figure, imshow(W);

title('Result watermark')

function [IW, capacity] = getContainer(I)

capacity = 0;

IW = I;

for i = 1:size(IW, 1)

for j = 1:size(IW, 2)

for k = 1:size(IW, 3)

IW(i,j,k) = bitand(IW(i,j,k), uint8(254));

capacity = capacity + 1;

end

end

end

capacity = capacity / 8;

end

## Реализация сокрытия данных в черных/белых пикселях

Рис. 5 – исходная вотермарка и полученная из изображения вотермарка

Суть алгоритма заключается в том, чтобы скрыть данные в черно/белых пикселях, которые не заметны человеческому глазу. Яркость таких пикселей меньше 16 или больше 239. В каждом таком пикселе можно скрыть 4 бита данных. Однако проблема заключается в том, что не всегда в изображении достаточно таких черно-белых пикселей (как это произошло у меня). В исходном изображении контейнер с таким алгоритмом мог хранить лишь 87 байт, чего недостаточно для вотермарки. Поэтому пришлось применить *imresize*. Коэффициент корреляции составил 0.9999.

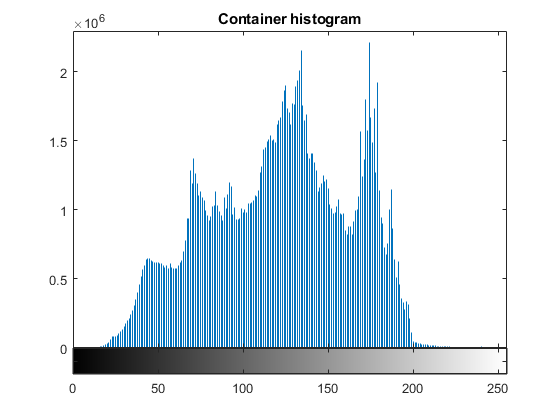
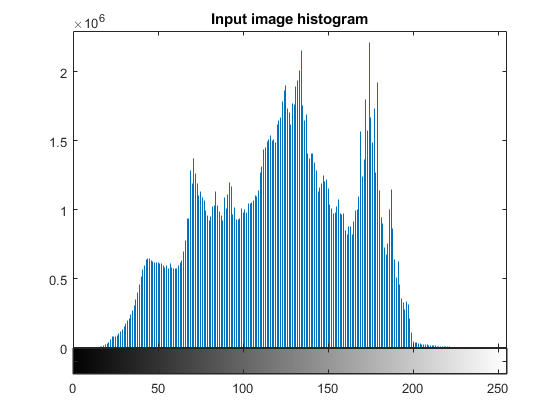


Рис. 6 – гистограммы исходного изображения с увеличением в 15 раз и гистограмма контейнера

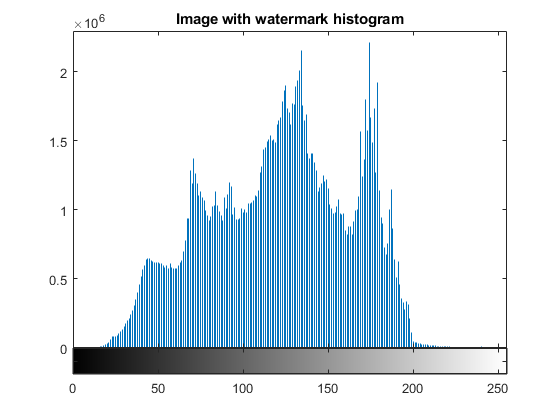


Рис. 7 – гистограмма изображения с вотермаркой

image = imread('resources/gray.bmp');

watermark = imread('resources/watermark.bmp');

image = imresize(image, 15); % to encrease the number of black/white pixels

figure, imshow(watermark)

title('Input watermark')

[container, capacity] = getContainer(image);

fprintf("Container capacity: %0.1f bytes\n", capacity);

assert(capacity >= numel(size(watermark)), "Not enough black/white pixels");

imageWithWatermark = setWatermark(container, watermark);

watermark = getWatermark(imageWithWatermark, size(watermark));

figure, imshow(watermark)

title('Result watermark')

figure, imhist(image)

title('Input image histogram')

figure, imhist(container)

title('Container histogram')

figure, imhist(imageWithWatermark)

title('Image with watermark histogram')

function [IW, capacity] = getContainer(I)

capacity = 0;

IW = I;

for i = 1:size(IW, 1)

for j = 1:size(IW, 2)

for k = 1:size(IW, 3)

byte = IW(i,j,k);

if (byte < uint8(16) || byte > uint8(239))

capacity = capacity + 1;

IW(i,j,k) = bitand(byte, 240);

end

end

end

end

capacity = capacity / 2; % only 4 bits are stored in black/white pixel

end

function IW = setWatermark(I, W)

n = 1;

m = 1;

is\_upper = 0;

IW = I;

for i = 1:size(IW, 1)

for j = 1:size(IW, 2)

for k = 1:size(IW, 3)

if (n == size(W, 1) && m == size(W, 2))

return;

end

byte = IW(i,j,k);

if (byte == 0 || byte == 240)

wbyte = uint8(W(n,m));

if (mod(is\_upper, 2) ==0)

wbyte = bitand(wbyte, 15); % lower 4 bits

is\_upper = 1;

else

wbyte = bitand(wbyte, 240);

wbyte = wbyte / 16;

is\_upper = 0;

m = m + 1;

if (m == size(W, 2) + 1)

n = n + 1;

m = 1;

end

end

IW(i,j,k) = byte + wbyte;

end

end

end

end

assert(n == size(W, 1) && m == size(W, 2), "Not enough dark and light pixels");

end

function W = getWatermark(IW, sizeW)

n = 1;

m = 1;

is\_upper = 0;

W = zeros(sizeW);

for i = 1:size(IW, 1)

for j = 1:size(IW, 2)

for k = 1:size(IW, 3)

if (n == size(W, 1) && m == size(W, 2))

return;

end

byte = IW(i,j,k);

if (byte < uint8(16) || byte > uint8(239))

if (mod(is\_upper, 2) ==0)

byte = byte \* 16;

byte = byte / 16;

W(n,m) = byte;

is\_upper = 1;

else

byte = byte / 16;

byte = byte \* 16;

W(n,m) = W(n,m) + byte;

is\_upper = 0;

m = m + 1;

if (m == size(W, 2) + 1)

n = n + 1;

m = 1;

end

end

end

end

end

end

end