

## СОДЕРЖАНИЕ

Введение . . . . .	6
1 Теоретические аспекты изучаемой темы . . . . .	7
1.1 Основные понятия . . . . .	7
1.2 Цифровые водяные знаки и цифровые отпечатки . . . . .	8
1.3 Обобщенные стеганографические методы . . . . .	10
1.4 Стегоанализ . . . . .	10
2 LSB . . . . .	12
2.1 Общие сведения . . . . .	12
2.2 Алгоритм . . . . .	13
2.3 Формат файла PNG . . . . .	15
2.4 Реализация LSB для контейнера PNG . . . . .	18
3 Скрытие в спектральной области . . . . .	21
3.1 Общие сведения . . . . .	21
3.2 Дискретное косинусное преобразование . . . . .	21
3.3 JPEG . . . . .	22

## ВВЕДЕНИЕ

Стеганография — это практика сокрытия сообщения внутри другого сообщения или физического объекта. Тогда как криптография скрывает содержимое сообщения, стеганография скрывает сам факт существования какого-либо сообщения.

Стеганография часто используется совместно с криптографией, дополняя ее. Стеганографические методы сокрытия сообщения снижают вероятность обнаружения самой передачи сообщения. Если сообщение к тому же зашифровано, то это обеспечивает еще большую защищенность.

В настоящее время наибольшее распространение получила цифровая стеганография. Особенностью цифровой стеганографии является сокрытие информации внутри других цифровых объектов, таких как текст, изображения, видео, аудио и другие. Со все большим возрастанием роли интернет-технологий в жизни человека значимость стеганографии также возрастает.

Области применения стеганографии включают в себя:

- а) Защита авторского права и DRM (digital rights management).
- б) Незаметная передача информации.
- в) Защита конфиденциальной информации от несанкционированного доступа.

Цифровая стеганография является молодым и бурно развивающимся направлением. В последние годы стеганография все больше находит применение в области защиты прав собственности на информацию.

В своей работе я хочу рассмотреть основные стеганографические алгоритмы, определить области их применения, достоинства и недостатки, а также провести анализ возможных уязвимостей этих алгоритмов.

# 1 Теоретические аспекты изучаемой темы

## 1.1 Основные понятия

а) Стеганографическая система (стегосистема) — совокупность методов и средств, используемых для создания скрытого канала для передачи информации. Основные требования, предъявляемые к стегосистеме:

1) Безопасность системы определяется секретностью ключа. Это означает, что даже если потенциальный враг представляет работу стеганографической системы и статистические характеристики сообщений и контейнеров, это не дает ему дополнительных преимуществ при выявлении наличия или отсутствия сообщения в конкретном контейнере.

2) При обнаружении противником наличия скрытого сообщения он не должен смочь извлечь сообщение до тех пор, пока он не будет владеть ключом.

3) Алгоритм сокрытия информации не нарушает ее целостность и аутентичность. В некоторых случаях дополнительно требуется, чтобы алгоритм обеспечивал целостность сообщения при деформации контейнера.

4) Система с цифровым водяным знаком должна иметь низкую вероятность ложного обнаружения скрытого сообщения.

б) Сообщение — информация, передачу которой нужно скрыть.

в) Контейнер — любая информация, используемая для сокрытия тайного сообщения. Контейнер может находиться в одном из двух состояний:

1) Пустой контейнер — контейнер, не содержащий сообщение.

2) Заполненный контейнер (стегоконтейнер) — контейнер, содержащий сообщение.

г) Стеганографический канал (стегоканал) — канал передачи стеко-контейнера.

д) Ключ (стегоключ) — секретный ключ, нужный для сокрытия стеко-контейнера. По аналогии с криптографией стегоключи подразделяются на 2 типа:

- 1) Закрытый стегоключ. В системах с закрытым стегоключем ключ должен быть создан до начала обмена сообщениями, либо передан по защищенному каналу связи.
- 2) Открытый стегоключ. Такой ключ может быть передан по открытому незащищенному каналу связи. Открытый стегоключ должен обладать таким свойством, чтобы по нему вычислительно нецелесообразно было восстанавливать закрытый ключ.

## 1.2 Цифровые водяные знаки и цифровые отпечатки

Цифровой водяной знак (ЦВЗ) — технология, созданная для защиты авторских прав на цифровые объекты. В связи с быстрым развитием информационных технологий все более актуальным становится вопрос защиты авторских прав и интеллектуальной собственности, представленной в цифровом виде. Примерами цифровых объектов могут выступать аудиозаписи, видеозаписи, изображения, электронные книги и другие. ЦВЗ могут быть как видимыми, так и невидимыми. Решение о наличии в цифровом объекте невидимого ЦВЗ принимаются на основе процедуры декодирования.

В общем виде стегосистема ЦВЗ может быть разбита на части следующим образом:

- а) Прекодер — часть, которая приводит ЦВЗ к удобному для встраивания в стегоконтейнер виду.
- б) Стегокодер — часть, которая вкладывает сообщение в стегоконтейнер.
- в) Выделение встроенного сообщения — процедура, выделяющая сообщение из стегоконтейнера.
- г) Стегодетектор — часть, определяющая наличие ЦВЗ.
- д) Декодер — часть, восстанавливающая исходное сообщение.

Контейнер, содержащий ЦВЗ, может подвергаться преднамеренным атакам или случайным помехам. Стегосистема ЦВЗ должна обеспечивать как различимость самого стегоконтейнера человеком, потому как в качестве стегоконтейнера выступает интеллектуальная собственность, направленная на конечного потребителя, так и различимость ЦВЗ стегодетекто-

ром, который может подтвердить или опровергнуть авторские права на интеллектуальную собственность. В связи с этим в стегосистемах ЦВЗ применяется помехоустойчивое кодирование и метод широкополосного сигнала.

Одной из основных характеристик ЦВЗ является надежность. Под надежность понимается устойчивость к различным деформациям контейнера. По отношению к этой характеристики ЦВЗ распадается на три класса:

а) Хрупкие. Такие ЦВЗ разрушается при небольших модификациях заполненного стегоконтейнера. Такие ЦВЗ применяются для аутентификации сигнала. Например, такие ЦВЗ используются для подтверждения подлинности цифрового объекта.

б) Полухрупкие. Такие ЦВЗ чувствительны к некоторым преобразованиям контейнера и нечувствительны к другим. Например, ЦВЗ, встроенное в изображение, может быть нечувствительно к его компрессии, но в то же время быть чувствительно к вырезке из этого изображения фрагмента.

в) Робастные или надежные. Такие ЦВЗ устойчивы к разным видам воздействия на контейнер. Такие ЦВЗ часто применяются при защите от копирования.

Чтобы осуществить вложение ЦВЗ в стегоконтейнер, ЦВЗ преобразуют к более удобному виду. Например, если ЦВЗ является изображением, то удобно будет представить его как двумерную битовую матрицу. Так же если ЦВЗ является изображением, разумно будет использовать не само изображение, а его Вайвлет преобразование или дискретное косинусное преобразование. Изображения обладают большой визуальной избыточностью. Данные преобразования концентрируют большую часть энергии (визуальной информации) в нижних частотах. Поэтому их можно использовать как низкочастотные фильтры. То же самое относится и к контейнерам.

Похожим на ЦВЗ, но отличающимся понятием является цифровой отпечаток. ЦВЗ предполагает встраивание одного и того же сообщения в различные контейнеры. В случае же цифрового отпечатка в каждый контейнер встраивается уникальное сообщение. Часто областью применения цифровых отпечатков становится защита исключительного права. В каче-

стве сообщения в таком случае встраивается информация, указывающая на идентифицирующие данные покупателя. Эти данные позволяют отследить источник распространения, если произойдет утечка стегоконтейнера.

### 1.3 Обобщенные стеганографические методы

К настоящему моменту разработано множество стеганографических методов скрытия информации. Для их систематичного изучения удобно группировать их по схожим признакам.

а) Пространственные методы. Особенностью этих методов является сокрытие информации напрямую в пространственной области контейнера. Например, в случае звукового контейнера таким пространством могут быть семплы, а в случае изображения — пиксели.

б) Частотные методы. Такие методы сначала используют одно из интегральных преобразований сигнала, чтобы перейти в его частотную область. Далее кодирование сообщение производится за счет изменения частотных характеристик сигнала. После этого используется обратное преобразование, чтобы получить модифицированный сигнал, содержащий закодированное сообщение.

в) Алгоритмы, использующие особенности формата файла. Такие алгоритмы как правило записывают сообщение в метаданные файла или в иные неиспользуемые поля файла.

### 1.4 Стегоанализ

Стегоанализ — это наука о выявлении сообщений, скрытых методами стеганографии. Задача стегоанализа — выявить подозрительные контейнеры, определить, есть ли в них скрытое сообщение, и, если возможно, восстановить это сообщение.

Если в случае криptoанализа аналитик начинает работу сразу с зашифрованным сообщением, то в случае стегоанализа аналитик начинает работу с множества подозрительных файлов, о которых как правило мало что известно. Деятельность аналитика в таком случае начинается с сокра-

щения этого множества файлов до подмножества, в котором файлы скорее всего были заполнены сообщением.

Основной техникой, используемой в стегоанализе, является статистический анализ. Сначала множество незаполненных контейнеров одного типа анализируется для получения различной статистики. Затем эта статистика используется при классификации контейнера как заполненного или пустого. При такой классификации могут быть использованы самые различные наблюдения:

- a) Сокрытие информации может приводить к изменению статистической структуры контейнера, в результате чего соседние элементы контейнера становятся попарно ближе друг к другу. На этом основана атака хиквадрат.
- б) Сокрытие информации увеличивает энтропию контейнера. В результате чего он хуже поддается сжатию. На этом основана атака с помощью алгоритмов сжатия.

Так же для такой классификации могут быть использованы методы машинного обучения.

## 2 LSB

### 2.1 Общие сведения

LSB (least significant bit) — стеганографический метод сокрытия информации, основанный на замене последних значащих бит элементов контейнера битами сообщения. Этот метод использует тот факт, что уровень детализации во многих контейнерах гораздо выше того, что может воспринять и различить человек. Следовательно, заполненный контейнер будет неотличим от оригинального для человеческого восприятия. В качестве примера можно взять полутоновое изображение с градациями серого. Цвет кодируется одним байтом. Человеческий глаз воспринимает только первые 7 байт, а самый младший бит вносит там мало информации, что человек не сможет заметить разницу.

LSB обладает следующими достоинствами:

- а) Простота реализации и эффективность.
- б) Низкая вычислительная сложность.
- в) Пустой и заполненный контейнер неразличимы для органов восприятия человека

И недостатками:

- а) Метод применим лишь к контейнерам, которые хранят данные без сжатия или используют сжатие без потерь, так как информация, закодированная в наименее значимых битах, может быть потеряна в процессе сжатия.
- б) Небольшие трансформации контейнера приводят к невозможности восстановить сообщение. Например, если сообщение скрыто в изображении методом LSB, то небольшие линейные трансформации (вращение, движение, отражение, гомотетия, сжатие, растяжение) уничтожают сообщение. Так же сообщение разрушается в результате сжатия с потерями. Все это говорит о том, что метод обладает низкой робастостью.
- в) Факт сокрытия изображения легко обнаруживает методами стеганализа.

Ввиду перечисленных выше недостатков очевидным кажется недопустимость использования данного методы для сокрытия ЦВЗ.

## 2.2 Алгоритм

Перейдем к конкретным реализациям этого метода. Алгоритм 1 демонстрирует псевдокод сокрытия методом LSB.

---

### Алгоритм 1: LSB Кодирование

---

**Data:** Контейнер, Сообщение

**Result:** Заполненный стегоконтейнер

$N \leftarrow$  Длина сообщения в битах;

$Message \leftarrow$  Бинарное представление сообщения;

$Container \leftarrow$  Массив с элементами контейнера;

**for**  $i = 1, 2, \dots, N$  **do**

**if**  $Container[i] \equiv Message[i] \pmod{2}$  **then**

**continue**;

**else**

$Container[i] \leftarrow (Container[i] \wedge \neg 1) \vee Message[i];$

---

### Алгоритм 2: LSB Декодирование

---

**Data:** Заполненный контейнер

**Result:** Сообщение в бинарном представлении

$Message \leftarrow$  Пустой список;

$Container \leftarrow$  Массив с элементами контейнера;

$N \leftarrow$  Длина  $Container$ ;

**for**  $i = 1, 2, \dots, N$  **do**

**if**  $Container[i] \equiv 0 \pmod{2}$  **then**

$Message.append(0);$

**else**

$Message.append(1);$

Как видно, сначала сообщение преобразуется в бинарный вид, а затем кодируется в элементах контейнера за счет изменения четности младшего бита. Логические операции в данном случае соответствуют бинарным операциям на компьютере. В итоге последние биты элементов контейнера в точности повторяют сообщение. Так же можно заметить, что

единственная часть алгоритма, зависящая от контейнера — это выделение массива элементов из контейнера. Алгоритм 2 показывает, как декодировать сообщение из заполненного стегоконтейнера.

Реализуем этот алгоритм в виде класса на Python. Как уже было сказано, существенная часть алгоритма не зависит от контейнера, поэтому целесообразно реализовать алгоритм как абстрактный класс, от которого будут наследоваться реализации для конкретных контейнеров.

### Листинг 2.1 — Абстрактный класс LSB

```
1 from abc import ABC, abstractmethod
2 from bitarray import bitarray
3
4 import numpy as np
5
6
7 class LSB(ABC):
8     def __init__(self, container: np.array, message = None):
9         """
10         Возвращает простой lsb кодер—,
11         принимает на вход контейнер и сообщение массив( байт) .
12         """
13
14     if message is None:
15         # По умолчанию сообщение пустое
16         self.message = []
17     else:
18         self.message = message
19
20     self.container = container
21
22     def encode(self):
23         """
24         Кодирует сообщение в контейнер
25         """
26
27         # Получаем последовательность элементов контейнера
28         elements = self._to_elements()
29
30         # Преобразуем сообщение к бинарному виду
31         np_message = np.unpackbits(np.frombuffer(self.message,
32                                     dtype=np.uint8)).ravel()
33
34         n = len(np_message)
35
36         # Меняем наименее значимый бит так,
37         # чтобы он кодировал биты сообщения
38         elements[:n] = (elements[:n] & ~1) | np_message
39
40         # Из элементов собираем контейнер обратно
41         self._from_elements(elements)
```

```

35
36     def decode(self):
37         """
38             Декодирует сообщение из контейнера
39         """
40
41         # Получаем последовательность элементов контейнера
42         elements = self._to_elements()
43
44         # Выбираем размер сообщения так, чтобы он был кратен размеру байта
45         size = len(elements) // 8 * 8
46
47         # Сообщение считываем из наименее значащих бит элементов контейнера
48         np_message = (elements[:size] & 1)
49
50         # Преобразуем битовую последовательность в байты
51         message = np.packbits(np_message.reshape(-1, 8), axis=-1).tobytes()
52         return message
53
54
55     @abstractmethod
56     def _to_elements(self):
57         """
58             Преобразует контейнер в последовательность элементов
59         """
60
61         pass
62
63     @abstractmethod
64     def _from_elements(self, elements):
65         """
66             Собирает контейнер из элементов
67         """
68
69         pass

```

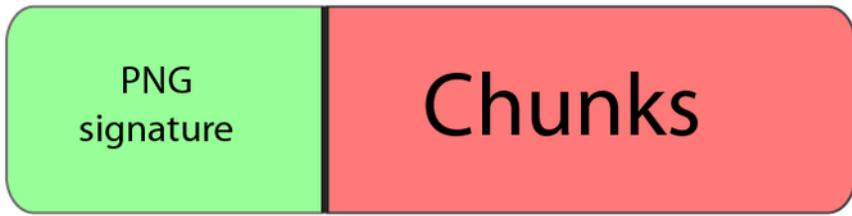
Как видно, в методах нет циклов **for**. Они скрыты за интерфейсом библиотеки `numtr`. Интерфейс библиотеки `numtr` позволяет нам применять операции к матрицам, из-за чего код выглядит лаконичнее. К тому же библиотека написана на языке C, поэтому ее код работает очень быстро.

Напишем реализацию LSB для PNG. Прежде чем реализовать метод LSB для PNG-контейнера, имеет смысл коротко изложить формат данных PNG.

### 2.3 Формат файла PNG

В самом общем виде PNG файл представляет из себя сигнатуру, за которой следует последовательность блоков, как показано на рисунке 2.1

Рисунок 2.1 — Общий вид формата PNG



Сигнатурой PNG файла состоит из 8 байт, в hex нотации они выглядят так: **89 50 4E 47 0D 0A 1A 0A**.

Каждый блок состоит из четырех секций: длина, тип, содержание, CRC, — как показано на рисунке 2.2: В длине указывается длина блока в

Рисунок 2.2 — Общий вид чанка

Length (длина)	Тип (имя) чанка	Содержание чанка	CRC
4 байта	4 байта	<i>Length</i> байт	4 байта

байтах. Тип указывается с помощью четырех ascii символов, чувствительных к регистру. С помощью регистра декодеру передает дополнительная информация, а именно:

- Регистр первого символа сообщает, является данный блок критическим или нет. Критические блоки распознаются каждым декодером. Если декодер не может распознать тип такого блока, он аварийно завершает работу.
- Регистр второго символа задает публичность или приватность блока. Публичные блоки обычно официальные и хорошо задокументированы. Чтобы закодировать в библиотека какую-то специфичную информацию, его тип можно изменить на приватный.
- Регистр третьего символа зарезервирован на будущее. По умолчанию там стоит символ в большом регистре.

г) Регистр четвертого символа сообщает возможность копирования данного блока редакторами.

Список критических блоков:

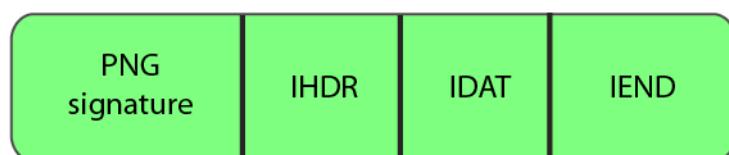
- а) IHDR — заголовочный блок, содержащий основную информацию об изображении.
- б) PLTE — палитра изображения.
- в) IDAT — содержит непосредственно изображение. В любом PNG файле должно быть не менее одного такого блока.
- г) IEND — завершающих чанк. Должен находиться в самом конце файла.

Список некритических блоков:

- а) bKGD — блок, задающий фоновый цвет изображения.
- б) cHRM — блок, используемый для задания цветового пространства CIE 1931.
- в) gAMA — определяет гамму.
- г) hIST — хранит гистограмму изображения либо общее содержания каждого цвета в рисунке.
- д) iTXT — содержит текст в UTF-8
- е) pHYs — содержит размер пикселя или отношение сторон изображения.
- ж) sRGB — свидетельствует об использовании sRGB схемы.
- и) tIME — дата последнего изменения изображения.
- к) tRNS — информация о прозрачности.

Согласно вышесказанному, минимальный PNG файл выглядит так, как показано на рисунке 2.3

Рисунок 2.3 — Минимальный PNG



В следующей секции представлены данные блока. В секции CRC записан CRC блока.

Наиболее интересными для нас являются блоки с типами IHDR и IDAT. IHDR — заголовочный блок, который является обязательным для PNG файла. Он содержит следующие интересующие нас поля:

- а) Ширина изображения в пикселях.
- б) Высота изображения в пикселях.
- в) Битовая глубина, задающее количество бит на каждый сэмпл.
- г) Тип цвета. Возможны следующие значения:
  - 1) Градация серого
  - 2) RGB
  - 3) Индексы из палитры
  - 4) Градация серого и альфа-канал
  - 5) RGB и альфа-канал

Блок IDAT содержит сжатые данные изображения. На данный момент поддерживается только сжатия по алгоритму deflate.

## 2.4 Реализация LSB для контейнера PNG

PNG изображение представимо в виде матрицы, элементами которой являются пиксели. В случае RGB каждый пикセル представляет элементы из трех каналов, каждый из каналов в отдельности может рассматриваться как градация серого. В случае градации серого каждый пикセル просто представлен значением от 0 до 255. Чтобы закодировать сообщение в эту матрицу, склеим ее строки друг с другом в одну большую строку, равно как и склеим каналы, чтобы они образовали последовательность элементов. Именно это делает метод `_to_elements`. Такой метод одинаково хорошо подходит и для разных типов цвета: RGB, RGB и альфа-канала, градации серого, градации серого и альфа-канала. Чтобы из элементов получить двумерную RGB матрицу, проделаем обратную операцию, что и делает метод `_from_elements`.

В функции `main` используем как сообщение книгу "Алиса в стране чудес" в оригиналe. Считаем файл с книгой как последовательность байт и закодируем в изображение с помощью LSB. Далее выполним декодиро-

вание и сверим полученные данные. Исходный код представлен в листинге 2.2.

### Листинг 2.2 — реализация LSB для PNG

```
1 from lsb import LSB
2 from PIL import Image
3
4 import numpy as np
5
6
7 class PNG(LSB):
8
9     def __init__(self, file_name: str, message: str):
10         self.file_name = file_name
11         container = np.array(Image.open(file_name))
12         super().__init__(container, message)
13
14     def _to_elements(self):
15         return self.container.ravel()[:]
16
17     def _from_elements(self, elements):
18         self.container.ravel()[:] = elements
19
20     def save(self):
21         image = Image.fromarray(self.container)
22         image.save(self.file_name)
23
24     def save_as(self, file_name):
25         image = Image.fromarray(self.container)
26         image.save(file_name)
27
28
29 def main():
30     with open("Messages/Alice in wonderland.txt", "rb") as f:
31         message = f.read()
32
33     size = len(message)
34     png = PNG("Images/Lenna.png", message)
35     png.encode()
36     png.save_as("Images/LSB_Lenna.png")
37     decoded = png.decode()
38
39     new_message = decoded[:size].decode()
40     print(f"new_message == message.decode() {new_message == message.decode()}")
41
42
```

```
43 | if __name__ == "__main__":
44 |     main()
```

Сравнение изображение до и после заполнения контейнера методом LSB приведено на рисунке 2.4. Как видно, два рисунка визуально неотличимы, хотя в одном из них закодировано 150 килобайт информации.



Рисунок 2.4 — Изображение до и после применения LSB

### 3 Сокрытие в спектральной области

#### 3.1 Общие сведения

#### 3.2 Дискретное косинусное преобразование

Дискретное косинусное преобразование (ДКП) — одно из дискретных преобразований Фурье. ДКП представляет конечную последовательность в виде суммы функций косинуса, колеблющихся на разных частотах. ДКП широко используется при обработке сигналов и сжатии данных. Например, ДКП используется при сжатии в изображениях (JPEG, HEIF), аудиофайлах (Dolby Digital, MP3), видеофайлах (MPEG, H.26x), в цифровом телевидении (SDTV, HDTV, VOD) и в других.

ДКП является линейным ортогональным преобразованием. Как любое дискретное линейное преобразование, ДКП можно представить в виде матрицы. Буду ортоганальным преобразованием, обратное к ДКП преобразование задает транспонированной матрицей ДКП, домноженной на какой-то коэффициент.

Использование косинусных, а не синусоидальных функций имеет решающее значение для сжатия, поскольку для аппроксимации типичного сигнала требуется меньше косинусных функций. ДКП подобно дискретному преобразованию Фурье, но использующее только действительные числа.

Существует 8 стандартных типов ДКП, однако наиболее употребимым является второй тип, который часто называют просто ДКП (DCT-II). Формула дискретного косинусного преобразования выглядит так, как показано в формуле 3.1:

$$X_k = \sum_{n=0}^{N-1} x_n \cos \left[ \frac{\pi}{N} \left( n + \frac{1}{2} \right) k \right] \quad k = 0, \dots, N-1 \quad (3.1)$$

Формула для матрицы преобразования выглядит как формуле 3.2:

$$DCT-2_n = \left[ \cos \left( k \left( l + \frac{1}{2} \right) \frac{\pi}{n} \right) \right]_{0 \leqslant k, l < n} \quad (3.2)$$

Как и в случае быстрого преобразования Фурье, существуют алгоритмы быстрого ДКП преобразования.

DCT-II часто используется для сжатия с потерями благодаря своему свойству уплотнения энергии: в типичных случаях большая часть информации, которую содержит сигнал, концентрируется в нескольких первых коэффициентах разложения.

Существуют так же многомерные ДКП, которые получаются из одномерных путем композиции ДКП по каждому измерению. Вывод такого преобразования для двумерного случая показан в формуле 3.3.

$$\begin{aligned} X_{k_1,k_2} &= \sum_{n_1=0}^{N_1-1} \left( \sum_{n_2=0}^{N_2-1} x_{n_1,n_2} \cos \left[ \frac{\pi}{N_2} \left( n_2 + \frac{1}{2} \right) k_2 \right] \right) \cos \left[ \frac{\pi}{N_1} \left( n_1 + \frac{1}{2} \right) k_1 \right] \\ &= \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x_{n_1,n_2} \cos \left[ \frac{\pi}{N_1} \left( n_1 + \frac{1}{2} \right) k_1 \right] \cos \left[ \frac{\pi}{N_2} \left( n_2 + \frac{1}{2} \right) k_2 \right] \end{aligned} \quad (3.3)$$

Здесь  $[x_{n_1,n_2}]$  — матрица до преобразования, и  $[X_{k_1,k_2}]$  — матрица после преобразования. В матричном виде это преобразование может быть представлено так, как показано в формуле 3.4, где  $x$  — матрица, которую нужно преобразовать.

$$X = (DCT\text{-}2_n)x(DCT\text{-}2_n^T) \quad (3.4)$$

Именно такое преобразование используется при компрессии в JPEG.

### 3.3 JPEG

JPEG является широко используемым методом сжатия с потерями для цифровых изображений. Степень сжатия регулируется, что позволяет выбирать между качеством и размером изображения. JPEG наиболее широко используемый стандарт сжатия изображений в мире и наиболее используемый формат цифровых изображений.

ДКП лежит в основе сжатия методом JPEG. Как уже говорилось выше, ДКП был выбран именно благодаря свойству уплотнения энергии. Чтобы прояснить, о чём идет речь, мной была сделана визуализация преобразования ДКП.

Выберем на изображении область 32x32 пикселя, как показано на рисунке 3.1.



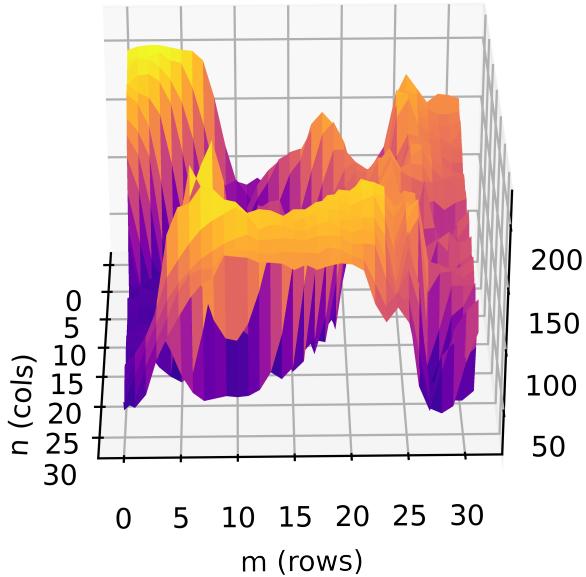
Рисунок 3.1 — Выбираем область

Сначала рассмотрим матрицу пикселей как двумерную дискретную функцию. Расположим координаты так, чтобы в левом верхнем углу располагался пиксель с координатами  $p_{k,j}$ ,  $k = 0, j = 0$ . Визуализацию можно посмотреть на рисунке 3.2.

Умножим эту функцию справа на транспонированную матрицу ДКП по формуле 3.4. Мы получим новую функцию, которая показана на рисунке 3.3. Таким образом фактически ДКП применилось к каждой строке матрицы. Из изображения видно, что наибольшие коэффициенты расположены в нижней части спектра, то есть ближе к нулевому столбцу.

К полученной матрице применим ДКП еще раз, в этот раз по столбцам. В полученной матрице наибольшее значение имеет коэффициент с координатами  $k = 0, j = 0$ . Этот коэффициент называется DC-коэффициент.

Рисунок 3.2 — Визуализация пикселей



Остальные коэффициенты называются АС-коэффициентами. Матрица показана на рисунке 3.4.

DC-коэффициент блока равен среднему всех пикселей в блоке, взято-му с определенным коэффициентом. Удаляя все коэффициенты, кроме DC, мы можем аппроксимировать блок пикселей их средним арифметическим. Чем дальше коэффициент располагается от DC, тем меньше психовизуаль-ной информации он несет для человека, и тем более незаметные детали изображения он хранит в себе.

Рисунок 3.3 — После применения ДКП к строкам матрицы

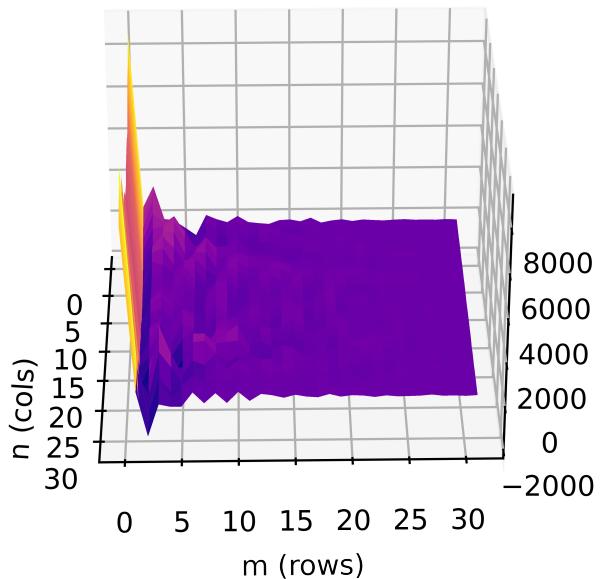


Рисунок 3.4 — После применения ДКП к строкам матрицы

