# Введение

Стеганография. История. Применение

# Вейвлет-преобразования

## Общая теория

## Вейвлет-преобразования в методах внедрения ЦВЗ в изображения.

### Что это такое

### Разновидности

### Возможные уязвимости

# Алгоритм внедрения цифрового водяного знака в изображение с помощью вейвлет-преобразований

Краткая сводка: алгоритм является неслепым, бла-бла-бла…

В реализованном мною алгоритме И.Р. Кима используется трехуровневое разложение с использованием вейвлет-преобразования Хаара. ЦВЗ встраивается в коэффициенты всех уровней разложения последовательно, начиная с третьего и заканчивая первым. Для встраивания выбираются коэффициенты, превышающие значение определенного порога.

Для встраивания используется аддитивный алгоритм

|  |  |
| --- | --- |
| , | (1) |

где – измененный коэффициент, – выбранный для внедрения коэффициент, – бит ЦВЗ, коэффициент масштабирования, который регулируется для каждого уровня разложения. Для получения изображения, содержащего ЦВЗ, используется обратное 2D-ДВП.

Для извлечения ЦВЗ необходимо исходное изображение. Применяется инверсия формулы внедрения, учитывая адаптивно-уровневый коэффициент масштабирования ,

|  |  |
| --- | --- |
|  | (2) |

Количество встраиваемой информации относительно невелико. В среднем оно составляет примерно 1/120 от размера исходного изображения, однако при расчете возможного количества встраиваемой информации следует учитывать не только размер изображения, но и насыщенность цвета определенных составляющих.

## Описание алгоритма И.Р. Ким и его программной реализации

### Встраивание ЦВЗ

Обобщенная блок-схема встраиваемых данных водяного знака с помощью 2D-ДВП изображена на рис. 1.



Рис. 1 Обобщенная блок-схема внедрения ЦВЗ

**Шаг 1.** Импортированиеисходного изображение и извлечение его синей составляющей, изменения в которой наименее заметны человеческому глазу.

**Шаг 2.** Импортирование ЦВЗ и преобразование его в одномерный массив бит.

**Шаг 3.** Разложение синей составляющей исходного изображения с помощью вейвлет-преобразования Хаара на 3 уровня и формирование коэффициентов аппроксимации и детализации.

**Шаг 4.** Вычисление порогов, в зависимости от которых будут выбраны пригодные для встраивания коэффициенты.

Пороги вычисляются отдельно для каждого уровня. На каждом уровне разложения находится абсолютный максимум для каждого набора коэффициентов, аппроксимации и детализации – горизонтальных , вертикальных и диагональных . Затем из них берется максимум . Порог вычисляется по следующей формуле:

|  |  |
| --- | --- |
|  | (5) |

**Шаг 5.** Процесс встраивания ЦВЗ. Начиная с 3-го уровня разложения, перебираются коэффициенты аппроксимации и детализации. Каждый коэффициент сравнивается со значением порога, соответствующего данному уровню. Коэффициенты, превышающие значение порога, изменяются в соответствии с формулой 1. Аналогичные операции производятся на 2 и на 1 уровне. В процессе выполнения этого шага функция периодически проверяет, сколько символов ЦВЗ осталось встроить. Если ЦВЗ встроено полностью, процесс останавливается. Соответственно, ЦВЗ не всегда встраивается во все коэффициенты.

**Шаг 6.** Восстановление синей составляющей изображения, в которое был встроен ЦВЗ, из измененных коэффициентов с помощью обратного вейвлет-преобразования. Изображение визуально ничем не должно отличаться от исходного.

### Извлечение ЦВЗ

**Шаг 1.** Выполнение шага 1 и шага 3 алгоритма встраивания к изображению со встроенным ЦВЗ. В результате чего получаем коэффициенты аппроксимации и детализации 3 уровней разложения Хаара.

**Шаг 2.** Выполнение шага 1 и шага 3 алгоритма встраивания к исходному изображению. Это действие необходимо в связи с тем, что алгоритм И.Р. Ким является неслепым, а это значит, что для извлечения ЦВЗ необходимо исходное изображение.

**Шаг 3.** Вычисление порогов для коэффициентов. Вычисление производится с помощью исходного изображения аналогично шагу 4 предыдущего алгоритма.

**Шаг 4.** Извлечение ЦВЗ. Для извлечения ЦВЗ производится перебор коэффициентов синей составляющей исходного изображения, начиная с 3 уровня. Если коэффициент больше порога , то находим соответствующий коэффициент в изображении со встроенным ЦВЗ и извлекаем бит ЦВЗ в соответствии с формулой 2.

**Шаг 5.** Перевод одномерного извлеченного массива бит ЦВЗ к двумерному виду и сохранение его в формате изображения. Для выполнения этого шага необходимо указать размеры исходного ЦВЗ.

## Программная реализация алгоритма И.Р. Ким

Программа, реализующая алгоритм И.Р., описанный в пункте 2.1, написана на языке C# на платформе .NET Framework. Реализовано 7 классов, код которых приведен в приложениях А-Д. Далее следует описание основной функциональности каждого класса.

### Хранение и обработка изображений

Для реализации хранения и обработки изображений было написано два класса. Вспомогательный класс DoublePixel (Приложение А) дает возможность оперировать значениями пикселей. Он содержит функции для быстрого доступа к конкретному пикселю, замены его значения и приведения пикселя к формату RGB.

Класс DoubleImage (Приложение Б) был реализован с целью замены стандартного класса для обработки изображений Bitmap. Он дает более быстрый доступ к отдельным пикселям в отличие от класса Bitmap за счет использования класса DoublePixel. Класс DoubleImage позволяет извлекать отдельные цветовые компоненты по выбору и обновлять их. Этот класс также содержит функцию для преобразования изображения в стандартный формат Bitmap для последующего сохранения.

Для работы ЦВЗ оказалось недостаточно реализованных классов для работы с изображением, поэтому потребовалось реализовать класс Watermark (Приложение В). Он позволяет преобразовывать изображение в двумерный массив бит и обратно.

### Вейвлет-преобразование

Для осуществления вейвлет-преобразования Хаара были реализованы два класса, HaarTransfrom (Приложение Г) и Wavelet (Приложение Д).

Класс HaarTransfrom содержит функцию Transform, принимающую на вход матрицу вещественных чисел, а также заданную ширину и высоту для преобразования отдельной части матрицы. Функция Transform сначала осуществляет преобразования по строкам матрицы, затем аналогично по столбцам. Для этого создается пустая матрица такого же размера для сохранения промежуточных результатов. В ходе преобразований по строкам для каждой строки *k* берется пара соседних чисел пробегает от 0 до *width/*2, где *width* – это ширина матрицы, с шагом 2, и преобразуется следующим образом:

|  |  |
| --- | --- |
|  | (3) |
|  | (4) |

Затем полученные значения аппроксимации *approx* и детализации *detal* сохраняются в промежуточную матрицу в строку *k* по индексам *i* и *width/2*, соответственно.

Преобразования по столбцам производятся на промежуточной матрице, полученной в результате преобразований по строкам. Преобразования производятся аналогично описанным выше преобразованиям по строкам. Результат сохраняется в конечную матрицу, которую возвращает функция Transform. Если преобразования производятся только на отдельной части матрицы, остальную часть марицы, полученной в результате преобразований, необходимо заполнить значениями из исходной матрицы. Для этого в классе HaarTransfrom используется вспомогательная функция FillRestMatrix.

Класс HaarTransfrom также содержит функцию Untransform, которая производит обратные преобразования для восстановления исходной матрицы.

Класс Wavelet содержит функцию Transform, которая принимает на вход матрицу вещественных чисел и необходимый уровень разложения. После проверки возможности разложения до указанного уровня запускается функция Transform класса HaarTransfrom. Размеры изменяемой части матрицы зависят от уровня разложения. Для первого уровня разложения требуется вся матрица, для последующих уровней размеры изменяемой части определяются следующим образом:

*variableWidth = width / i,*

*variableHeight = height / i,*

где *width и height –* исходные размеры изображения, а *i –* текущий уровень разложения.

Класс Wavelet содержит функцию Untransform, которая производит обратное вейвлет-преобразование.

Также класс Wavelet содержит функцию GetCoefficient, которая возвращает матрицу коэффициентов (аппроксимации и детализации – горизонтальных, вертикальных и диагональных) в зависимости от требуемого уровня разложения.

### Алгоритмы встраивания и извлечения ЦВЗ

Алгоритмы по встраиванию извлечению ЦВЗ реализованы в классе JRKimAlgorithm (Приложение Е). Также реализован вспомогательный класс Threshold (Приложение Ж) для подсчета порогов.

Функция KIMembed осуществляет встраивание ЦВЗ в изображение. Она получает на вход исходное изображение и ЦВЗ. Далее извлекается синяя составляющая исходного изображения с помощью функции GetColorComponent класса DoubleImage. Затем вызывается функция Transform класса Wavelet с синей составляющей и значением 3 уровня разложения в виде параметров. Затем все коэффициенты разложения извлекаются из полученной в результате разложения матрицы с помощью функции GetCoefficient класса Wavelet и сохраняются в отдельные двумерные массивы.

Затем необходимо получить значения порогов. Для этого используется функция GetTreshold класса Treshold. Функция GetTreshold перегружена и может принимать на вход 3 или 4 матрицы в зависимости от уровня разложения, поскольку на 1 и 2 уровнях разложения используются только коэффициенты детализации, а на 3 уровне еще и коэффициенты аппрокисмации.

После получения порогов ЦВЗ преобразуется с помощью функции TransformWatermark. Ее назначение поменять все нулевые биты ЦВЗ на значение -1 для осуществления дальнейших арифметических операций.

Затем происходит последовательное встраивание ЦВЗ в коэффициенты, начиная с коэффициентов аппроксимации с 3 уровня разложения и заканчивая коэффициентами детализации 1 уровня разложения.

После чего собирается новая матрица с измененными коэффициентами с помощью функции SetCoefficient класса Wavelet. К ней применяется обратное вейвлет-преобразования Хаара с помощью функции Untransfrom класса Wavelet. В результате получается измененная синяя составляющая, которая несет в себе код ЦВЗ. Далее синяя составляющая исходного изображение заменяется на измененную с помощью функции UpdateColorComponent класса DoubleImage.

Функция KIMextract класса JRKimAlgorithm осуществляет извлечение ЦВЗ с помощью исходного изображения. Ее алгоритм описан в пункте 2.1.2. Данная функция использует аналогичные вспомогательные функции как в функции KIMembed.

### Пользовательская часть программы

Здесь возможно стоит, а возможно и не стоит описать, что происходит со стороны пользователя и какими средствами оно реализовано.

## Анализ алгоритма

Анализ на быстродействие. На робастность (сжатие, обрезка, поворот, масштабирование)

## Оптимизация алгоритма

### Встраивание ЦВЗ в несколько цветовых компонент

### Предварительный расчет максимально возможного и оптимального размера ЦВЗ

### Вывод предупреждения о невозможности встроить ЦВЗ целиком

### Алгоритм определения схожести восстановленного ЦВЗ в случае потери данных

# Заключение

# Список литературы