ОБРАБОТКА ИЗОБРАЖЕНИЙ, РАСПОЗНАВАНИЕ ОБРАЗОВ

ПОИСК ДУБЛИКАТОВ НА ЦИФРОВЫХ ИЗОБРАЖЕНИЯХ

Глумов Н.И.[1], Кузнецов А.В.[1], Мясников В.В.[1,2]
[1] Институт систем обработки изображений РАН,
[2] Самарский государственный аэрокосмический университет имени академика С.П. Королёва (национальный исследовательский университет)

Аннотация

В статье предлагаются два алгоритма поиска на изображениях неискажённых дубликатов – полностью совпадающих прямоугольных фрагментов. Оба алгоритма используют представление данных фрагмента в виде значения хэш-функции, но используют различные математические принципы для её построения: теоретико-вероятностный и теоретико-числовой. В работе представлено сравнение предложенных алгоритмов, а также даны рекомендации по их применению.

Ключевые слова: цифровое изображение, проверка подлинности, дубликат, хэш-таблица, абсолютные частоты, китайская теорема об остатках.

Введение

Цифровые изображения являются одним из важных способов представления визуальной информации. В современном мире, наряду с развитием высокотехнологичного программного обеспечения, появляется всё больше возможностей подделать цифровое изображение, например, в корыстных целях. Будем называть способы фальсификации изображений атаками.

Одной из самых часто применяемых атак является копирование области цифрового изображения из одной части в другую(-ие). Такие атаки будем называть дублированием фрагментов цифрового изображения, а сами копируемые фрагменты – дубликатами. Между копированием и вставкой дубликат может подвергаться искажениям, которые можно разделить на яркостные (добавление шума, линейное контрастирование, фильтрация) и геометрические (масштабирование, поворот, аффинное преобразование). В настоящее время существует целое направление, связанное с разработкой методов и алгоритмов поиска дубликатов на изображениях [1, 2, 3]. В то же время задачу поиска дубликатов (в различных её постановках) рассматривать как решённую не удаётся. В частности, как указано в работах [1, 5], до настоящего времени ни один из существующих алгоритмов не даёт полной гарантии обнаружения неискажённых дубликатов, то есть скопированных фрагментов, в которых никаких изменений не было произведено. Настоящая работа посвящена решению указанной задачи.

1. Понятие дубликата и общее описание предлагаемого алгоритма

Определим цифровое изображение размера $M\times N$ как отображение вида:

$$f:\mathbf{N}_M\times\mathbf{N}_N\to\{0,8\}^8.$$

Будем говорить, что на изображении присутствуют дубликаты с параметрами a, b, если существует по крайней мере две пары координат (m,n) и (m',n'), для которых следующие фрагменты изображений совпадают по значениям:

$$f(m,n),\ f(m,n+1),...,\ f(m,n+b-1),$$
$$...$$
$$f(m+a-1,n),..,\ f(m+a-1,n+b-1)$$

$$f(m',n'),\ f(m',n'+1),...,\ f(m',n'+b-1),$$

и ...

$$f(m'+a-1,n'),...,\ f(m'+a-1,n'+b-1).$$

Таким образом, в рамках настоящей работы в качестве претендентов на дубликаты мы рассматриваем только прямоугольные области.

Под *поиском дубликата с параметрами* a, b будем понимать задачу указания для каждого отсчёта (m,n), определяющего начало фрагмента изображения размера $a\times b$:

$$f(m,n),\ f(m,n+1),...,\ f(m,n+b-1),$$
$$...$$
$$f(m+a-1,n),..,\ f(m+a-1,n+b-1),$$

уникального номера $t(m,n)\in\mathbf{N}$, характеризующего фрагмент следующим образом:

$$t(m,n)\equiv\begin{cases}0, & \text{нет дубликата},\\ >0, & \text{номер «типа» дубликата.}\end{cases}$$

Под номером «типа» дубликата понимается некоторый уникальный номер, который оказывается одинаковым для совпадающих фрагментов (различающиеся фрагменты имеют различные номера).

В настоящее время существует де-факто стандарт построения алгоритмов поиска дублирующей информации, основанный на использовании хэш-функций [1, 2]. Для целей поиска дубликатов на изображении этот подход рассматривался во многих работах и предполагал разбиение изображения на множество частично перекрывающихся блоков [1, 2, 4], для каждого из которых рассчитывается одно или несколько значений хэш-функций. Результатом такого решения является невозможность полу-

чения полной гарантии нахождения всех дубликатов (положение блока может не совпасть с положением дубликата).

В настоящей работе предлагается проводить анализ абсолютно всех фрагментов изображения размера $a \times b$ последовательно – в режиме так называемого «скользящего окна» [6]. Для каждого положения «скользящего окна» по соответствующим отсчётам изображения вычисляется значение хэш-функции, и в соответствующую ему ячейку хэш-таблицы помещается абсолютная частота появления этого значения (фактически хэш-таблица представляет собой гистограмму значений хэш-функции). Абсолютные частоты со значением большим «1» соответствуют тем значениям хэш-функции, которые получены от потенциальных дубликатов (их положение на изображении указывается во время выполнения второго прохода изображения). Вычислительная сложность такого решения (просмотр абсолютно всех фрагментов изображения) нивелируется в настоящей работе способом построения и способом реализации расчёта хэш-функции (в силу тривиальности обобщения алгоритма на случай использования нескольких хэш-функций он в работе не рассматривается).

Таким образом, вместо попарного сравнения всех множеств пикселей $f(m,n)$ предлагается выявление совпадающих фрагментов изображения с помощью хэш-таблицы, содержащей абсолютные частоты появления хэш-значений.

Ниже предложены два способа определения и расчёта хэш-функции $\mathbf{H(i,j,}f\mathbf{)}$, приводящие к двум различным алгоритмам.

2. Хэш-функции и их обоснование

2.1. Хэш-функция, использующая битовое представление изображения

Рассмотрим использование в качестве значения хэш-функции целочисленную величину, бинарное представление которой есть последовательно (в рамках некоторого структурного элемента) взятые биты анализируемого фрагмента изображения $f(m,n)$. В идеальном случае следовало бы построить взаимно однозначное преобразование между значениями фрагмента изображения и значениями хэш-функции, например, интерпретируя последовательность бит всех пикселей окна обработки как целое число. В результате такого преобразования диапазон возможных значений составит $B = 2^{p \cdot a \cdot b}$ (p – количество бит в пикселе). Будем называть B *информационной ёмкостью* пикселей окна обработки. Это значение определяет длину хэш-таблицы (гистограммы). Тогда, например, для 8-битного изображения и окна размером 5×5 пикселей получаем $B = 2^{200} \approx 10^{60}$. Видно, что построение гистограммы такой длины физически нереализуемо даже при малых окнах. В таких условиях предлагается отказаться от требования взаимной однозначности преобразования, но обеспечить крайне малую вероятность случайного совпадения значений хэш-функции для несовпадающих окон изображения – *коллизий*.

Пусть с использованием некоторого преобразования изображение $f(m,n)$ приведено к бинарному $b(m,n) \in \{0,1\}$, удовлетворяющему условию: соседние отсчёты являются независимыми и одинаково распределёнными случайными величинами, с равной вероятностью принимающими значения $\{0,1\}$.

Сформулируем требования к хэш-функции, которая должна обеспечить реализацию поставленной задачи:

1) информационная ёмкость значений хэш-функции не должна превышать максимально возможной длины гистограммы $B = 2^k$, где k определяется из внешних условий, к которым можно отнести: размер «окна обработки» $a \times b$, размер доступной оперативной памяти для приложения, размер доступной оперативной памяти для конкретного вычислительного устройства (например, $k < 32$ для вычислительных систем ×86); дополнительные ограничения на k можно получить, если учесть специфику анализируемого изображения;
2) плотность распределения величин B должна быть близка к равномерной;
3) информационная ёмкость должна быть больше размеров анализируемого изображения $B > M \times N$, иначе даже при равномерном распределении будет большое количество ложных совпадений;
4) математическое ожидание числа ложных совпадений на анализируемом изображении не превышает некоторого заданного $E\mu_{(>\lambda)}$.

Пусть далее $T = 2^{k-1}$ – длина гистограммы значений хэш-функции, MN – число вычисляемых значений хэш-функции для изображения с линейными размерами $(M + a - 1) \times (N + b - 1)$. Тогда, используя известное решение классической задачи о «распределении частиц по ящикам» [7], можно показать, что математическое ожидание числа отсчётов гистограммы, в которые попали ровно r значений хэш-функции, составляет:

$$\mathrm{E}\mu_r = T \cdot C_{MN}^r \cdot \frac{1}{T^r} \cdot \left(1 - \frac{1}{T}\right)^{MN-r}.$$

Рассматривая ситуацию, когда конкретное значение хэш-функции встречается в гистограмме только один раз, то есть $r = 1$, получаем:

$$\mathrm{E}\mu_1 = MN\left(1 - \frac{1}{\Phi}\right)^{MN-1},$$

что одновременно соответствует числу положений «скользящего окна», для которых значения хэш-функций различаются, то есть анализируемые фрагменты изображения не являются дубликатами. Следовательно, математическое ожидание количества положений окна на обрабатываемом изображении, для которых выявляются ложные совпадения, имеет вид:

$$\mathrm{E}\mu_{(>1)} = MN - MN\left(1 - \frac{1}{\Phi}\right)^{MN-1}.$$

Используя неравенство Бернулли, можно получить следующее соотношение:

$$\mathrm{E}\mu_{(>1)} \le \frac{(MN)^2}{T}.$$

Это соотношение будет являться пятым требованием к разрабатываемой хэш-функции.

Из этого неравенства можно легко определить либо ограничения на максимальный размер обрабатываемого изображения, либо требования к размеру гистограммы (числу анализируемых бит в «скользящем окне»). Например, при $\lambda = 1$ опускаем всего одну ошибку – случайное совпадение значений хэш-функции всего для одной пары различающихся фрагментов) и $T = 2^{32}$ получаем $MN \le 2^{16} = 2^8 2^8$, т.е. предлагаемый алгоритм даст обозначенный выше уровень ошибок только изображениям с размером менее 256×256 пикселей. Для более крупных изображений необходимо либо значительно увеличивать T, либо допускать большее число ошибок $E\mu_{(>\lambda)}$.

Пусть результатом вычисления хэш-функции является битовая последовательность длиной kR (т.е. $B = 2^{kR}$, где $\mathrm{R} \in \mathbf{Z}$). Тогда если возможно разделить её на независимые подпоследовательности длиной k бит, то можно построить набор гистограмм Hst_r, $0 \le r < R$, по которому окончательно построить выходное поле, значения которого указывают на возможное наличие дубликатов.

Необходимое число подпоследовательностей легко определить следующим образом:

$$R \ge \left\lceil \frac{1}{k} \log_2 \left(\frac{(MN)^2}{\mathrm{E}\mu_{(>\lambda)}} \right) \right\rceil.$$

Таким образом, учитывая случайный независимый характер появления коллизий, удаётся уменьшить результирующее число коллизий до приемлемого уровня за счёт R-кратной обработки изображения.

Предлагаемый алгоритм обладает вычислительной сложностью порядка $O(u(ab)MNR)$, где $u(ab)$ – функция, зависящая от способа реализации хэш-функции (как правило, значения $u(ab)$ пропорциональны ab, однако при рекурсивной реализации могут быть постоянными и не зависеть от размеров окна ab).

В рамках настоящей работы мы рассмотрим два варианта определения бинарного изображения $b(m,n)$:

а) $f(m,n)$ – значение конкретного бита (битовая плоскость) или нескольких битов, объединённых в одну последовательность, величины $f(m,n)$. Примером является использование в качестве $b(m,n)$ значения младшего бита величины $f(m,n)$;

б) $b(m,n)$ – результат применения операции XOR ко всем битам пикселей $f(m,n)$.

2.2. Хэш-функция, использующая теоретико-числовое представление кода анализируемого фрагмента

Пусть «окно обработки» имеет размер $1 \times b$, тогда $f(m,n)$, $f(m,n+1)$, ... , $f(m,n+b-1)$ – анализируемый фрагмент (обобщение на случай, когда $a > 1$, тривиально и для краткости изложения не рассматривается) и $f(m,n) \in \lfloor 0, 2^8 - 1 \rfloor$. Рассматривая эти отсчёты совместно, мы можем взаимно-однозначно связать их со значением следующей хэш-функции:

$$H(m,n,f) \equiv f(m,n) \cdot 2^{8(b-1)} + \\ + f(m,n+1) \cdot 2^{8(b-2)} + \ldots + f(m,n+b-1) \cdot 2^0. \quad (1)$$

Для двумерного случая хэш-функция (1) является обобщением кольцевого хэша Рабина Карпа [8].

Проблемами такого представления в ПЭВМ являются:

– отсутствие стандартных типов данных для работы с большими числами,
– невозможность выделения памяти требуемых размеров для работы с хэш-таблицей соответствующего размера.

Мы предлагаем решение, основанное на использовании *китайской теоремы об остатках* (КТО) [9]. В основе теоремы лежит утверждение о том, что любое число x в диапазоне от 0 до $b_0 b_1 \ldots b_{R-1}$ можно представить (и восстановить, при необходимости) в виде остатков от деления его на взаимно простые числа $b_0, \ldots, b_{R-1}$:

$$x \equiv r_0 \bmod b_0$$
$$x \equiv r_1 \bmod b_1$$
$$\ldots$$
$$x \equiv r_{R-1} \bmod b_{R-1}.$$

Тогда любая из R функций вида

$$H_r(m,n,f) \equiv H(m,n,f) \mod b_r, \quad r = \overline{0, R-1} \quad (2)$$

может использоваться в качестве хэш-функции значения (1). Более того, вся система из R функций (2) гарантированно даёт взаимнооднозначное соответствие с анализируемым фрагментом, что позволяет при необходимости строить алгоритм с последовательным улучшением решения.

Рассмотрим общий вид функции $H_r(m,n,f)$:

$$H_r(m,n,f) = \Big(\big(f(m,n) \bmod b_r \cdot 2^{8(b-1)} \bmod b_r \big) \bmod b_r + \\ + \big(f(m,n+1) \bmod b_r \cdot 2^{8(b-2)} \bmod b_r \big) \bmod b_r + \\ + \ldots + f(m,n+b-1) \bmod b_r \Big) \bmod b_r .$$

Для вычисления значения данного выражения следует учесть следующие упрощения:

1) $f(m,n) \bmod b_r = f(m,n)$, ввиду ограничения $f(m,n) \in \lfloor 0, 2^8 - 1 \rfloor$ и того, что $b_r >> 2^8 - 1$;
2) нет необходимости вычислять для каждого положения окна обработки значения $q_r^i = 2^{8i} \bmod b_r$, $i \in [1, b-1]$ – достаточно вычислить их один раз перед проходом по изображению;
3) вычисление $\bmod b_r$ для каждого слагаемого может быть опущено с учётом того, что вычисляемая сумма в окне не будет превышать размерность разрядной сетки вычислительного средства (положим, что $b_r = 2^{31} - 1$, тогда порядок значения суммы в $H_r(m,n,f)$ будет равен $ab \cdot 2^8 2^{31}$, что является

допустимым значением для хранения в типе *long* при условии, что $ab < 2^{24}$).

С учётом вышеперечисленного, выражение для вычисления $H_r(m,n,f)$ упростится до следующего:

$$H_r(m,n,f) = \begin{pmatrix} f(m,n)q_r^{b-1} + \\ + f(m,n+1)q_r^{b-2} + \ldots \\ + f(m,n+b-1) \end{pmatrix} \bmod b_r. \qquad (3)$$

Заметим, что в силу особенностей представления чисел на современных ПЭВМ *обязательным ограничением на выбор чисел* $b_0, \ldots, b_{R-1}$ *является* $b_r \neq 2^s$ ($s \in \mathbf{N}$) (например, удобно использовать $b_r = 2^{31}-1$).

Также заметим, что для «скользящего» окна операция (1) как в исходном, так и в модулярном представлении (3) может выполняться рекурсивно, а именно:

$$H(m,n,f) = 2^8\left(H(m,n-1,f) - 2^{8(b-1)} f(m,n)\right) + \\ + f(m,n+b-1).$$

При этом умножение на степень «2» вычислительно эффективно реализуется на ПЭВМ путём операций регистрового сдвига. Это радикально сокращает время работы предложенного алгоритма расчёта хэш-функции.

3. Экспериментальные исследования разработанных хэш-функций

Ключевыми параметрами предложенных алгоритмов обнаружения дубликатов являются:

- хэш-функции (в работе предложены три варианта её определения: п.2.1.а, п.2.1.б, п.2.2);
- размер хэш-таблицы/длина гистограммы $T = 2^{k-1}$;
- параметры окна обработки *a* и *b*.

Показателем качества конкретного алгоритма (для конкретного набора изображений) является число ложно найденных дубликатов – коллизий (*K*). На графиках будем показывать относительное количество коллизий, то есть $\chi = K / MN$.

Для проведения экспериментов мы использовали три изображения, полученных со спутника SPOT-4, размером 5000×7000, дубликаты на которых отсутствовали. Для выполнения измерений использовался стандартный ПК (Core 2 Quad Q8300, 4 Gb RAM) с 64-битной ОС Windows 8, среда разработки MS Visual Studio 2012, среда исполнения – .NET 4.0.

Для простоты будем обозначать хэш-функцию и алгоритм на основе проецирования бит – 1 или ♦ (п.2.1.а), на основе XOR – 2 или ▲ (п.2.1.б), на основе КТО – 3 или ■ (п.2.2).

На рис. 1 показана зависимость χ от размера окна анализа. Как видно из графика, наилучший результат показывает хэш-функция 3, самой плохой является хэш-функция 2.

На рис. 2 показана зависимость χ от количества бит *k*, используемых для представления значений хэш-функции. Как видно из графика, для всех значений хэш-функции сохраняется тенденция увеличения χ с уменьшением *k*. Следует отметить, что для хэш-функции 3 увеличение числа коллизий происходит медленнее, чем для двух других.

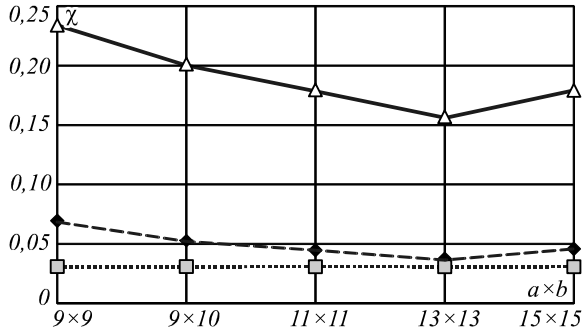


Рис. 1. Зависимость относительного количества коллизий от размера окна анализа

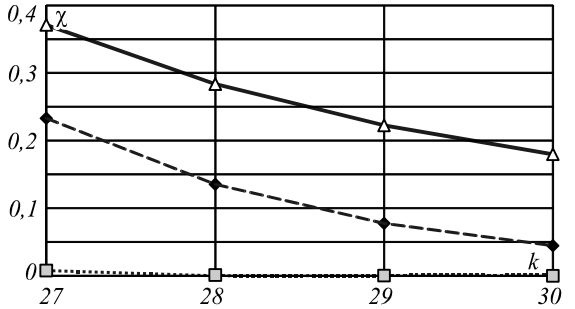


Рис. 2. Зависимость относительного количества коллизий от количества бит значений хэш-функции

Под *относительным временем* вычисления хэш-функции будем понимать величину:

$$\upsilon = \frac{t - \min}{\max - \min},$$

где *t* – среднее время (мс) вычисления хэш-функции по всем трём изображениям; min, max – минимальное и максимальное значения *t* среди трёх алгоритмов.

Результат зависимости υ от размера окна обработки представлен на рис. 3. Из графика видно, что хэш-функция 3 выполняется быстрее функции 1, а также время её работы не зависит от размера окна обработки ввиду использования рекурсивной реализации. Следует отметить, что представленные хэш-функции сравнимы по абсолютному времени выполнения (0 на графике соответствует 12 с, 1 – 19 с).

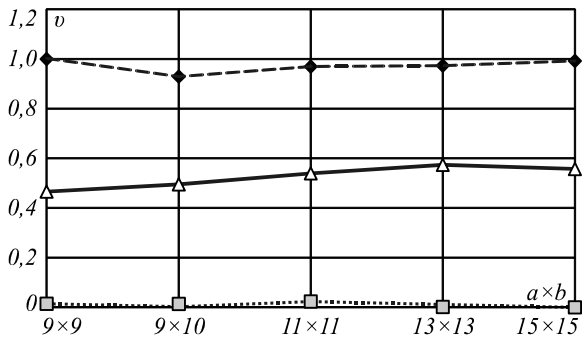


Рис. 3. Зависимость относительного времени работы хэш-функции от размера окна анализа

4. Алгоритмы поиска дубликатов на изображении

Предложенные хэш-функции фрагментов изображения можно использовать для поиска дубликатов различными способами. Ниже представлены два предлагаемых способа, учитывающие специфику алгоритмов вычисления значений хэш-функции.

4.1. Алгоритм поиска дубликатов, основанный на хэш-функции, проецирующей биты изображения

Построение алгоритма обнаружения дубликатов на основе хэш-функции, проецирующей биты изображения, основано на выборе структурного элемента, в рамках которого производится выбор бит для формирования хэш-значений. Как было сказано ранее, последовательность бит может включать все биты анализируемого изображения, расположенные последовательно друг за другом от младшего к старшему. В таком случае для каждого положения окна обработки последовательность будет состоять из $8ab$ бит, тогда максимальное количество подпоследовательностей или гистограмм значений хэш-функции легко вычислить как $R=\lfloor 8ab/k \rfloor$.

В качестве структурного элемента предлагается выбирать биты из последовательности длиной $8ab$ с периодом $P=\lfloor ab/k \rfloor$. В таком случае на некоторых итерациях алгоритма для формирования значения хэш-функции будут использоваться биты на стыке соседних битовых плоскостей. Выражение для вычисления позиции p в окне обработки в таком случае будет выглядеть следующим образом:

$$p=\left\lfloor \frac{r}{P} \right\rfloor Pk + r \bmod P + zP,\ r\in[0,R],\ z\in[0,k-1], \quad (4)$$

где r – номер итерации, z – порядковый номер бита хэш-значения, начиная от младшего.

Перед запуском алгоритма обнаружения формируется массив для хранения результирующего поля $t(i,j)$. На первом шаге при $r=0$ строится хэш-таблица/гистограмма $H_0(m,n,f)$, позиции бит изображения, используемых для построения хэш-значения, вычисляются по формуле (4). Одновременно с заполнением хэш-таблицы производится заполнение значениями поля $t(i,j)$. На следующем шаге при $r=1$ производится анализ только тех положений окна обработки, в которых, возможно, находятся дубликаты, то есть с учётом вычисленных значений $t(i,j)$ при $r=0$. На каждой итерации алгоритма количество коллизий уменьшается.

Итеративный процесс останавливается в случае, когда количество коллизий на итерации r совпадает с количеством коллизий на итерации $r-1$ либо пока коллизий не будет обнаружено.

В псевдокоде описанный алгоритм для итерации r выглядит следующим образом:

```
FOR i=0 to M–1
  FOR j=0 to N–1
    IF (t(i,j) != 0) THEN
      Hash(i,j)=Hr(i,j,f);
      Hist(Hash(i,j))++;
    END IF
  END
END
FOR i=0 to M–1
  FOR j=0 to N–1
    IF (Hist(Hash(i,j)) < 1) THEN
      t(i,j) = 0;
    ELSE
      t(i,j) = Hash(i,j);
    END IF
  END
```

Представленный алгоритм использует тот факт, что для вычислений хэш-функции на шагах $r>0$ нет необходимости производить вычисления для всего изображения.

4.2. Алгоритм обнаружения дубликатов, основанный на хэш-функции на базе КТО

Будем использовать для построения алгоритма одно значение b_0, например, $b_0=2^{31}-1$. Так как одной итерации недостаточно для обеспечения малого количества коллизий, будем заменять значение одной хэш-функции, вычисленной для окна обработки с размером $a\times b$, r значениями хэш-функции, вычисленной для окна обработки меньшего размера $a_0\times b_0$. Чтобы определить максимальные размеры окна обработки, которое может r раз с перекрытием поместиться в окне с размером $a\times b$, воспользуемся следующими формулами:

$$a_0=a-\left\lfloor \sqrt{r} \right\rfloor,\ \ b_0=b-\left\lfloor \frac{r}{a_0} \right\rfloor.$$

Далее будем строить хэш-таблицу/гистограмму для фрагментов размером $a_0\times b_0$ и формировать результирующее поле $t(i,j)$. После этого для положения окна обработки с размером $a\times b$ будем анализировать r значений поля $t(i,j)$: дубликат может находиться только в том положении окна, где все r значений поля $t(i,j)$ принимают ненулевые величины. Разработанный алгоритм в псевдокоде выглядит следующим образом:

```
FOR i=0 to M–1
  FOR j=0 to N–1
    FOR k=0 to r–1
      i1 = k/ a0 ;
      j1 = k% a0 ;
      IF (t(i, j) == 0)
        break;
      END IF
    END
    IF (k == r)
      result[i, j] = t(i, j);
    END IF
  END
END
```

5. Экспериментальные исследования разработанных алгоритмов

Для сравнения качества и времени работы алгоритмов обнаружения были построены зависимости относительного количества коллизий и относительного времени работы от длины хэш-таблицы и размера окна обработки. Результаты вычислений представлены на рис. 4, 5 и 6.

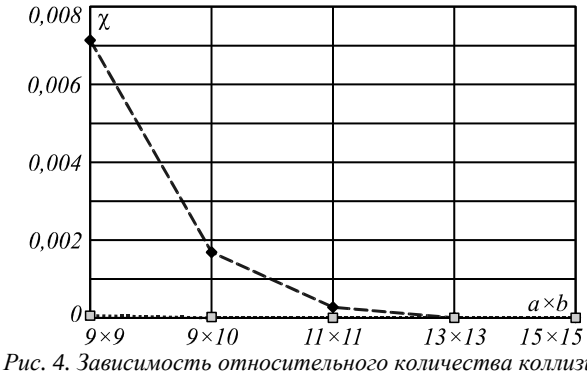


Рис. 4. Зависимость относительного количества коллизий от размера окна обработки

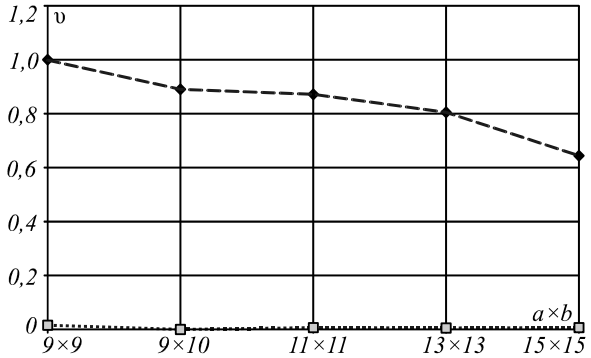


Рис. 5. Зависимость относительного времени работы алгоритмов от размера окна обработки

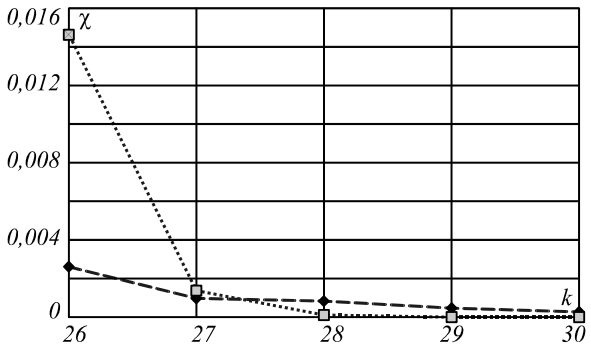


Рис. 6. Зависимость относительного количества коллизий алгоритмов от количества бит значений хэш-функции

Из рис. 4 можно сделать вывод, что алгоритм 3 превосходит алгоритм 1 – среднее относительное количество коллизий для алгоритма 3 составляет $5 \cdot 10^{-5}$.

Из рис. 5 можно видеть, что время работы алгоритма 3 не изменяется с увеличением размера окна обработки ввиду рекурсивного способа реализации, а время работы алгоритма 1, наоборот, уменьшается. Последнее связано с тем, что бо́льшие временные затраты приходятся на выполнение первой итерации, а так как количество возможных положений фрагментов (и, как следствие, коллизий) уменьшается с увеличением размера окна обработки (как видно из рис. 4), то последующие итерации алгоритма до момента его остановки выполняются быстрее. Иными словами, повышается скорость сходимости алгоритма.

На рис. 6 представлены результаты работы алгоритмов обнаружения дубликатов в разрезе «относительное количество коллизий – количество бит значений хэш-функции». Для проведения эксперимента использовалось окно обработки с фиксированным размером 11×11, для алгоритма 3 было выбрано $r = 4$. Наилучший результат показал алгоритм 3 при $k \geq 28$, при $k < 28$ алгоритм 1 обнаружил меньшее количество коллизий. Следует отметить, что уменьшать количество коллизий для алгоритма 3 можно путём увеличения r – время работы при этом будет меняться незначительно.

На рис. 7 представлена зависимость значения χ от значений r алгоритма 3 для окна обработки с размером 11×11.

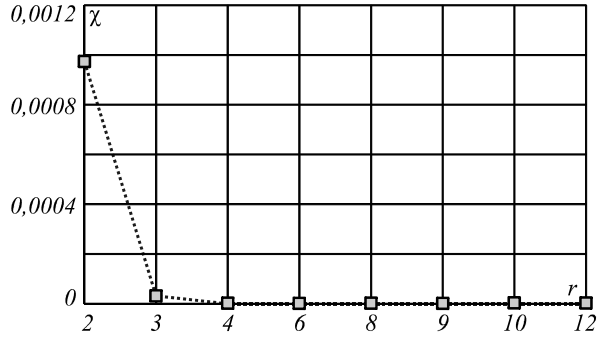


Рис. 7. Зависимость относительного количества коллизий от r для алгоритма на основе КТО

Из графика видно, что при $r = 4$ количество коллизий становится достаточно малым для того, чтобы можно было проверить соответствующие им положения окна обработки попиксельно на соответствие, снизив тем самым число коллизий до 0.

Выводы и рекомендации

В данной работе были предложены два алгоритма поиска на цифровых изображениях неискажённых дубликатов на основе трёх предложенных хэш-функций. Также было представлено сравнение предложенных алгоритмов. Дальнейшие исследования будут посвящены как улучшению (по качеству и времени) разработанных алгоритмов, так и разработке алгоритмов обнаружения на изображении искажённых дубликатов.

Благодарности

Работа выполнена при частичной финансовой поддержке:

– грантов РФФИ, проекты № 13-07-12103-офи-м, 13-01-12080-офи-м, 12-07-00021-а;

– программы фундаментальных исследований Президиума РАН «Фундаментальные проблемы информатики и информационных технологий», проект 2.12;

– Министерства образования и науки Российской Федерации (в рамках постановления Правительства Российской Федерации от 09.04.2010 г. № 218: договор № 02.Г36.31.0001 от 12.02.2013).

Литература

1. **Christlein, V.** An Evaluation of Popular Copy-Move Forgery Detection Approaches / V. Christlein, C. Riess, J. Jordan, C. Riess, E. Angelopoulou // IEEE Transactions on information forensics and security. – 2012. – Vol. 7, Issue 6. – P. 1841-1854.
2. **Farid, H.** Image Forgery Detection / H. Farid // IEEE Signal processing magazine. – 2009. – P. 16-25.
3. **Глумов, Н.И.** Обнаружение дубликатов на изображениях / Н.И. Глумов, А.В. Кузнецов // Компьютерная оптика. – 2011. – № 35(4). – С. 508-512.
4. **Fridrich, J.** Detection of copy-move forgery in digital images / J. Fridrich, D. Soukal // Proceedings of Digital Forensic Research Workshop. – 2003. – P. 55-61.
5. **Sridevi, M.** Comparative Study of Image forgery and Copy-move Techniques / M. Sridevi, C. Mala, S. Sanyam. – Proceedings of the Second International Conference on Computer Science, Engineering and Applications (ICCSEA 2012). – New Delhi, India, 2012. – P. 715-723.
6. Методы компьютерной обработки изображений / М.В. Гашников, Н.И. Глумов, Н.Ю. Ильясова, В.В. Мясников [и др.]; под общей ред. В.А. Сойфера. – 2-е изд., испр. – М.: Физматлит, 2003. – 784 с.
7. **Колчин, В.Ф.** Случайные размещения / В.Ф. Колчин, Б.А. Севастьянов, В.П. Чистяков. – М.: Наука, 1976. – 224 с.
8. **Биркгоф, Г.** Современная прикладная алгебра / Г. Биркгоф, Т. Барти. – М.: Мир, 1976. – 400 с
9. **Лидл, Р.** Конечные поля / Р. Лидл, Г. Нидеррайтер. – М.: Мир, 1988. – 820 с.

References

1. **Christlein, V.** An Evaluation of Popular Copy-Move Forgery Detection Approaches / V. Christlein, C. Riess, J. Jordan, C. Riess, E. Angelopoulou // IEEE Transactions on information forensics and security. – 2012. – Vol. 7, Issue 6. – P. 1841-1854.
2. **Farid, H.** Image Forgery Detection / H. Farid // IEEE Signal processing magazine. – 2009. – P. 16-25.
3. **Glumov, N.I.** Detection of Copy-Move Forgery Detection / N.I. Glumov, A.V. Kuznetsov // Computer optics. – 2011. – Vol. 35(4). – P. 508-512. – (In Russian).
4. **Fridrich, J.** Detection of copy-move forgery in digital images / J. Fridrich, D. Soukal // Proceedings of Digital Forensic Research Workshop. – 2003. – P. 55-61.
5. **Sridevi, M.** Comparative Study of Image forgery and Copy-move Techniques / M. Sridevi, C. Mala, S. Sanyam. – Proceedings of the Second International Conference on Computer Science, Engineering and Applications (ICCSEA 2012). – New Delhi, India, 2012. – P. 715-723.
6. Methods of computer image processing / M.V. Gashnikov, N.I. Glumov, N.U. Ilyasova, V.V. Myasnikov [et al]. – 2-nd edition reviewed. – Moscow: “Fizmatlit” Publisher, 2003. – 784 p. – (In Russian).
7. **Kolchin, V.F.** Random placements / V.F. Kolchin, B.A. Sevastyanov, V.P. Chistyakov. – Moscow: “Nauka” Publisher, 1976. – 224 p. – (In Russian).
8. **Birkhoff, G.** Modern applied algebra / G. Birkhoff, T. Bartee. – Moscow: “Mir” Publisher, 1976. – 400 p. – (In Russian).
9. **Lidl, R.** A classical introduction to modern number theory / R. Lidl, H. Niederreiter. – Moscow: “Mir Publisher”, 1988. – 820 p. – (In Russian).

THE ALGORITHM FOR COPY-MOVE DETECTION ON DIGITAL IMAGES

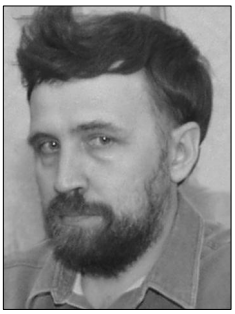
N.I. Glumov[1], A.V. Kuznetsov[1], V.V. Myasnikov[1,2]
[1] Image Processing Systems Institute of the RAS,
[2] S.P. Korolyov Samara State Aerospace University (National Reseach University)

Abstract

In the paper, we propose two algorithms for undistorted copy-move regions’ (coincident rectangular fragments) detection on digital images. Both algorithms represent pixels’ data as the hash function value, which is built using two different mathematical principles: probability-theoretical and number-theoretical. The comparison of the proposed algorithms, as well as recommendations for their use is also proposed in this paper.

Key words: digital image, authentication, copy-move region, hash table, absolute frequencies, chinese remainder theorem.

Сведения об авторах



Глумов Николай Иванович, родился в 1962 году. В 1985 году окончил Куйбышевский авиационный институт (ныне Самарский государственный аэрокосмический университет (национальный исследовательский университет)). В 1994 году защитил диссертацию на соискание степени кандидата технических наук. В настоящее время работает старшим научным сотрудником в Институте систем обработки изображений РАН. Круг научных интересов включает обработку изображений и распознавание образов, компрессию изображений, моделирование систем формирования цифровых изображений. Имеет свыше 100 публикаций, в том числе более 40 статей, две монографии (в соавторстве).

E-mail: *nglu@smr.ru* .

Nikolay Ivanovich Glumov (b. 1962) graduated with honours (1985) from the S. P. Korolyov Kuibyshev Aviation Institute (presently, S. P. Korolyov Samara State Aerospace University (SSAU)). He received his Candidate in Technics

(1994) degree from Samara State Aerospace University. He is the senior research scientist at the Samara Image Processing Systems Institute of the Russian Academy of Sciences (IPSI RAS). His current research interests include image processing and pattern recognition, images compression, digital images forming systems modelling. He has more than 100 publications, including more than 40 scientific papers, 2 monographs (in co-authorship).

Кузнецов Андрей Владимирович, родился в 1987 году. В 2010 году окончил Самарский государственный аэрокосмический университет (СГАУ) с отличием по специальности «Прикладная математика и информатика». В настоящее время работает стажёром-исследователем в Институте систем обработки изображений РАН, является аспирантом СГАУ. Круг научных интересов включает обработку и анализ изображений, обнаружение локальных изменений на изображениях, распознавание образов, геоинформатику. Имеет 20 публикаций, в том числе 5 научных статей.

E-mail: *kuznetsoff.andrey@gmail.com* .

Andrey Vladimirovich Kuznetsov (b. 1987) graduated with honours (2010) from the S. P. Korolyov Samara State Aerospace University (SSAU), majoring in Applied Mathematics and Informatics. He works as a researcher in Samara Image Processing Systems Institute of the Russian Academy of Sciences (IPSI RAS), also studies as a postgraduate student in SSAU. His research interests are currently focused on image processing and analysis, local images changes detection, pattern recognition, geoinformatics. He has 17 publications, including 4 scientific papers.

Мясников Владислав Валерьевич, 1971 года рождения. В 1994 году окончил Самарский государственный аэрокосмический университет (СГАУ). В 1995 году поступил в аспирантуру СГАУ, в 1998 году защитил диссертацию на соискание степени кандидата технических наук, а в 2008 – диссертацию на соискание степени доктора физико-математических наук. В настоящее время работает ведущим научным сотрудником в Федеральном государственном бюджетном учреждении науки Институте систем обработки изображений РАН и одновременно профессором кафедры геоинформатики и информационной безопасности СГАУ. Круг научных интересов включает цифровую обработку сигналов и изображений, компьютерное зрение, распознавание образов, искусственный интеллект и геоинформатику. Имеет более 100 публикаций, в том числе 40 статей и две монографии (в соавторстве). Член Российской ассоциации распознавания образов и анализа изображений.

E-mail: *vmyas@smr.ru* . Страница в интернете: *http://www.ipsi.smr.ru/staff/MyasVV.htm* .

Vladislav Valerievich Myasnikov (1971 b.), graduated (1994) from the S.P. Korolyov Samara State Aerospace University (SSAU). He received his PhD in Technical sciences (2002) and DrSc degree in Physics & Maths (2008). At present he is a leading researcher at the Image Processing Systems Institute of the Russian Academy of Sciences and holds a part-time position of Associate Professor at the Department of Geoinformatics and Information Security at SSAU. The area of interests includes digital signals and image processing, geoinformatics, neural networks, computer vision, pattern recognition and artificial intelligence. He's list of publications contains about 100 scientific papers, including 40 articles and 2 monographs. He is a member of Russian Association of Pattern Recognition and Image Analysis.

Поступила в редакцию 27 июня 2013 г.