УДК 004.932.72

Д. В. ЗАЕРКО, В. А. ЛИПНИЦИКИЙ

АЛГОРИТМ ВЕСОВОГО ОПРЕДЕЛЕНИЯ ГРАНИЧНЫХ ПИКСЕЛЕЙ

Белорусский государственный университет информатики и радиоэлектроники, Минск, Республика Беларусь

При работе с методами подавления цифрового шума, основанных на операций двумерной свертки, возникает необходимость обхода алгоритмами граничных пикселей в пиксельной матрице полутонового изображения. Проблема возникает в связи с особенностью самого алгоритма свертки, по которому происходит воздействие центра ядра матрицы свертки к элементу пиксельной матрицы. Данная особенность характерна для целого класса методов, использующих операцию двумерной свертки. Существует ряд примитивных способов ее решения, однако, ни один из этих способов не соблюдает консенсус между экономным использованием ресурсов и заполнением граничных пикселей кодом (числом бит на пиксел) полутона, наиболее близким по полутонам с соседними пикселями. Объект исследования в статье – пиксельная матрица полутонового изображения. Предмет изучения – алгоритмы заполнения граничных пикселей близким кодом полутона при воздействии ядра на пиксельную матрицу полутонового изображения. Основная цель – создание эффективного алгоритма заполнения граничных пикселей матрицы близкими по коду полутона с соседними для последующего использования этих граничных значений при выполнении двумерной свертки. Заполненные граничные пиксели позволят учесть в операции свертки все пиксели исходного полутонового изображения. Предлагается алгоритм заполнения проблемного граничного пикселя на этапе пошагового вычисления величины влияния центра ядра свертки на пиксел, при обращении к которому алгоритм свертки выходит за границы пиксельной матрицы оригинального полутонового изображения. Алгоритм учитывает «особые» случаи выхода за границы и позволяет при обращении к несуществующему элементу определить код полутона пикселя. Алгоритм прост для программирования и легко интегрируется с базовым алгоритмом работы двумерной свертки в методах подавления цифрового шума.

Ключевые слова: *Пиксельная матрица, дефекты цифрового изображения, подавление цифрового шума, операция двумерной свертки, шумофильтрация, граничные пиксели, полутоновые изображения.*

Введение

Пиксельная матрица, с точки зрения компьютерной обработки, является главным объектом изучения и содержит всю полноту информационных характеристик растрового изображения. Логично, что при обработке исходной пиксельной матрицы машина работает лишь с численными значениями и не имеет понятия об эффектах, критически важных для восприятия человеческим глазом: эффектах отсутствия четкости, цветовых дефектах, наличии цифрового шума и т. д. Все эти дефекты цифрового изображения должны быть устранены на этапе предварительной обработки. Данный этап предполагает получение на основе оригинала максимально точного и адаптированного для автоматического анализа изображения. От его выполнения существенно зависит сохранение или изменение полноты информационных характеристик анализируемого изображения.

Предварительная обработка изображения [1–5] для подавления цифрового шума, как основного побочного эффекта, возникающего при работе с фото-сенсорами и электронными устройствами, предполагает использование математических методов работы с матрицами линейного усреднения точек по соседям и т. д. Связующим звеном в этих методах, кроме использования пиксельной матрицы как объекта преобразования, является использование алгоритма двумерной свертки. Однако, использование данного алгоритма сопряжено с проблемой вычисления значений сверки на граничных пикселях матрицы растрового изображения [6]. Рассмотрим подробнее эту проблему и предложим метод ее решения.

Операция двумерной свертки, проблема граничных пикселей

В практике цифровой обработки изображений широко используется некаузальная

масочная фильтрация. Маской называют функцию $a(i, j)$, которая представляет собой весовые коэффициенты, заданные во всех точках окрестности, обычно симметрично окружающую рабочую точку кадра. В теории масочной фильтрации часто используется понятие двумерной свертки, с помощью которой можно создавать графические фильтры и воздействовать на изображения. Как известно, в работе с изображениями, свёртка – это операция вычисления нового значения заданного пикселя, при которой учитываются значения окружающих его соседних пикселей [7, с. 47]. Главным элементом свёртки является **ядро свёртки,** представляющее собой матрицу произвольного размера и отношения сторон (чаще всего матрица размером 3x3). Основная идея работы свертки проста [8, 9]. При вычислении нового значения выбранного пикселя изображения, ядро свёртки воздействует своим центром ядра на пиксел изображения. Соседние пиксели так же взаимодействуют с ядром. Далее, вычисляется сумма произведений значений пикселей изображения на значения, сопоставляемого с данным пикселем элемента ядра. Полученная сумма и является новым значением выбранного пикселя. Если применить двумерную свёртку к каждому пикселю изображения, то получится некий эффект, зависящий от выбранного ядра свертки. А теперь зададимся вопросом, как должен отработать алгоритм свертки ~~на~~ краях изображения? Существует множество ответов на этот вопрос, однако оптимального, с точки зрения объемов использования памяти и уменьшения шума на преобразованном изображении, без потери основных дескрипторов, пока не найдено. Наиболее часто используются следующие подходы.

1. Создание изображения большего размера, чем исходное, у которого на краях будут заданы дополнительные значения пикселей. Дополнительные значения могут быть: равными 0, полученными с другой стороны изображения, продублированным крайним пикселем изображения, отраженными относительно границ, полученными экстраполяцией и т. д.

2. Создание промежуточного изображения. В центр изображения копируется входная картинка, а края заполняются крайними пикселями изображения. Размытие применяется к промежуточному буферу, а потом из него извлекается результат.

Это лишь немногие методы решения возникшей проблемы. Учитывая все возрастающий интерес к визуальному распознаванию и коррекции растровых изображений, однозначно можно утверждать то, что число разнообразных фильтров на основе операции двумерной свертки будет лишь увеличиваться. В свою очередь, это приводит к поиску более эффективных способов решения проблемы граничных пикселей в алгоритме двумерной свертки. Применение различных методов для обработки одного и того же изображения предполагает неодинаковое использование вычислительных ресурсов. Основной же целью является улучшение качества обрабатываемого изображения. Это указывает на необходимость соблюдения консенсуса между экономией вычислительных ресурсов и улучшением обрабатываемого изображения. Алгоритм свертки, дополненный модификацией для корректной обработки граничных пикселей, не должен расточительно использовать ресурсы, но определять наиболее близко код полутона граничных пикселей относительно исходного изображения.

Весовой алгоритм определения кода полутона для граничных пикселей полутонового изображения

На наш взгляд, наиболее эффективным здесь является следующий, альтернативный, весовой метод работы с кодами полутона граничных пикселей для алгоритма двумерной свертки, основные черты которого были впервые представлены в [10]. Модификация не предполагает создания промежуточного изображения, а лишь использования квадратной подматрицы пикселей $A^+ = \{a^+_{i,j}\}_{n\times n}$ порядка n, состоящей из n^2 пикселей оригинальной матрицы $A = \{a_{I,J}\}_{h\times w}$; $n \le \min(h+2\, w+2)$. Номера строк и столбцов матрицы A обозначим через $I = 0, h+1, J = 0, w+1$.

Очень важны диапазоны изменения индексов строк и столбцов i, j (далее так же k, p) элементов пиксельной подматрицы. Диапазоны изменений будут существенно разниться, в зависимости от расположения стороны (левая, нижняя, правая, верхняя)

граничных пикселей относительно оригинальной пиксельной матрицы полутонового изображения A. Расположение и соответствующие ему диапазоны индексов будут определяться, исходя из заданных номеров строк и столбцов $I=\overline{0,h+1}, J=\overline{0,w+1}$ граничного элемента матрицы требующего вычисления по правилу:

$$(i,j)=\begin{cases} i=\overline{1+n(I-1),In}; j=\overline{1,n}; & если, \quad J=0; 0<I<h+1; \\ i=\overline{(h+1)-n,h}; j=\overline{1+n(J-1),Jn}; & если, \quad 0<J<w+1; I=h+1; \\ i=\overline{1+n(I-1),In}; j=\overline{(w+1)-n,w}; & если, \quad J=w+1; 0<I<h+1; \\ i=\overline{1,n}; j=\overline{1+n(J-1),Jn}; & если, \quad 0<J<w+1; I=0. \end{cases}$$

Далее указание изменения диапазонов строк и столбцов в описании формул будет опускаться, предполагая выбор его заранее, в зависимости от $I=\overline{0,h+1}, J=\overline{0,w+1}$.

Код полутона проблемного пикселя определяется, исходя из наиболее часто встречающихся кода полутонов среди других t^2 пикселей $a^+_{i,j}$ подматрицы A^+ и располагается в непосредственной близости от граничного пикселя.

Под близкими, имеется в виду t^2 пикселей, являющихся смежными к граничному пикселю на глубину до t пикселей. Величина t определяется произвольно, исходя только из трех условий: четного значения, ограничений вычислительных возможностей системы и размера изображения, ограниченных меньшей границей т.е $t=\begin{cases} 3,5,..,h-1; & h\le w; \\ 3,5,...,w-1; & w\le h. \end{cases}$.

Модификация предполагает использование специальных весовых коэффициентов пикселей $\alpha_{i,j}$ для каждого пикселя, отвечающего за оценку «встречаемости» этого кода полутона среди других кодов полутонов t^2 пикселей.

Итак, опишем основные этапы алгоритма, учитывая известные строку и столбец $I=\overline{0,h+1}, J=\overline{0,w+1}$ граничного пикселя, для которого необходимо провести вычисление.

1. Вычисление угловых точек. Особым случаем будет расчёт 4 «угловых» граничных пикселей относительно оригинального полутонового изображения, которые могут быть вычислены сразу же, и из-за их небольшого числа не будут играть существенную роль.

$$a_{0,0}=a^+_{1,1}; a_{h+1,0}=a^+_{h,1}; a_{0,w+1}=a^+_{1,w}; a_{h+1,w+1}=a^+_{h,w}.$$

2. Подготовка весовых коэффициентов пикселей. Все весовые коэффициенты $\alpha_{i,j}$ для элементов $a^+_{i,j}$ принимают равные значения $\alpha_{i,j}=\frac{1}{t^2}$. Значение t выбирается, исходя их двух приведенных выше ограничений.

3. Вычисление результирующих весовых коэффициентов. Для вычисления результирующего весового коэффициента $res(\alpha_{i,j})$ для пикселя $a^+_{i,j}$ необходимо суммировать весовые коэффициенты $\alpha_{k,p}$; для пикселей $a^+_{k,p}$ при $a^+_{i,j}=a^+_{k,p}$. Иначе, оставить результирующий вес $res(\alpha_{i,j})$ неизменным, то есть $res(\alpha_{i,j})=\sum_k\sum_p\{\alpha_{k,p}\mid a^+_{i,j}=a^+_{k,p}\}$.

Определение значения кода полутона граничного пикселя. Значения граничных пикселей определяется по оценке результирующих весовых коэффициентов, полученных на шаге 2.

Среди результирующих весовых коэффициентов $res(\alpha_{i,j})$ для пикселей $a^+_{i,j}$ находится коэффициент с максимальным значением. Если его значение не равно заданному значению на первом шаге: $\max\{res(\alpha_{i,j})\}\neq\frac{1}{t^2}$, то очевидно, что коэффициент характеризует пиксель $a^+_{i,j}$ с наиболее часто повторяемым полутоном среди других t^2 пикселей, и граничный пиксель примет его значение. Если максимальное значение не изменилось: $\max\{res(\alpha_{i,j})\}=\frac{1}{t^2}$, то все t^2 пиксели имеют различные коды полутона или сформировались группы пикселей, равные по числу пикселей, но различные по коду полутона. В этом случае граничным пикселем будет первый ближайший смежный пиксель, то есть: $a^+_{i,1}$ для левых граничных пикселей на изображении, $a^+_{i,w}$ для правых, $a^+_{1,j}$ для верхних, $a^+_{h,i}$ для нижних. Представим вышесказанное в виде формул, подставляя вместо $I=\overline{0,h+1}, J=\overline{0,w+1}$ номер строки и столбца для пикселя, для которого необходимо провести вычисление.

Для угловых граничных пикселей справедливы следующие выражения 1–4:

При $J=0;1\le I<\left\lfloor\frac{t}{2}\right\rfloor+1;$ или $J=0;h-\left\lfloor\frac{t}{2}\right\rfloor<I\le h$ для левых граничных пикселей

$$a_{I,0}=\begin{cases}a^{+}_{i,j},если \quad \max\{res(\alpha_{i,j})\}\ne\frac{1}{t^2};i=\overline{1,\left\lfloor\frac{t}{2}\right\rfloor+1};i=\overline{h-\left\lfloor\frac{t}{2}\right\rfloor,h};j=\overline{1,\left\lfloor\frac{t}{2}\right\rfloor};\\ a^{+}_{i,1},если \quad \max\{res(\alpha_{i,j})\}=\frac{1}{t^2};i=\overline{1,\left\lfloor\frac{t}{2}\right\rfloor+1};i=\overline{h-\left\lfloor\frac{t}{2}\right\rfloor,h};j=\overline{1,\left\lfloor\frac{t}{2}\right\rfloor}.\end{cases}\tag{1}$$

При $I=h+1;1\le J<\left\lfloor\frac{t}{2}\right\rfloor+1;$ или $I=h+1;h-\left\lfloor\frac{t}{2}\right\rfloor<J\le h;$ для нижних граничных пикселей:

$$a_{h+1,J}=\begin{cases}a^{+}_{i,j},если \quad \max\{res(\alpha_{i,j})\}\ne\frac{1}{t^2};i=\overline{h-\left\lfloor\frac{t}{2}\right\rfloor,h};j=\overline{1,\left\lfloor\frac{t}{2}\right\rfloor+1};j=\overline{w-\left\lfloor\frac{t}{2}\right\rfloor,w};\\ a^{+}_{i,1},если \quad \max\{res(\alpha_{i,j})\}=\frac{1}{t^2};i=\overline{h-\left\lfloor\frac{t}{2}\right\rfloor,h};j=\overline{1,\left\lfloor\frac{t}{2}\right\rfloor+1};j=\overline{w-\left\lfloor\frac{t}{2}\right\rfloor,w}.\end{cases}\tag{2}$$

При $J=w+1;1\le I<\left\lfloor\frac{t}{2}\right\rfloor+1;$ или $J=w+1;w-\left\lfloor\frac{t}{2}\right\rfloor<I\le w;$ для нижних граничных пикселей:

$$a_{I,w+1}=\begin{cases}a^{+}_{i,j},если \quad \max\{res(\alpha_{i,j})\}\ne\frac{1}{t^2};i=\overline{1,\left\lfloor\frac{t}{2}\right\rfloor+1};i=\overline{h-\left\lfloor\frac{t}{2}\right\rfloor,t};j=\overline{w-\left\lfloor\frac{t}{2}\right\rfloor,w};\\ a^{+}_{i,1},если \quad \max\{res(\alpha_{i,j})\}=\frac{1}{t^2};i=\overline{1,\left\lfloor\frac{t}{2}\right\rfloor+1};i=\overline{h-\left\lfloor\frac{t}{2}\right\rfloor,t};j=\overline{w-\left\lfloor\frac{t}{2}\right\rfloor,w}.\end{cases}\tag{3}$$

При $I=0;1\le J<\left\lfloor\frac{t}{2}\right\rfloor+1$ или $I=0;w-\left\lfloor\frac{t}{2}\right\rfloor<J\le w$ для нижних граничных пикселей:

$$a_{0,J}=\begin{cases}a^{+}_{i,j},если \quad \max\{res(\alpha_{i,j})\}\ne\frac{1}{t^2};i=\overline{1,\left\lfloor\frac{t}{2}\right\rfloor+1};j=\overline{1,\left\lfloor\frac{t}{2}\right\rfloor+1};j=\overline{w-\left\lfloor\frac{t}{2}\right\rfloor,w};\\ a^{+}_{i,1},если \quad \max\{res(\alpha_{i,j})\}=\frac{1}{t^2};i=\overline{1,\left\lfloor\frac{t}{2}\right\rfloor+1};j=\overline{1,\left\lfloor\frac{t}{2}\right\rfloor+1};j=\overline{w-\left\lfloor\frac{t}{2}\right\rfloor,w}.\end{cases}\tag{4}$$

При $J=0;\left\lfloor\frac{t}{2}\right\rfloor<I\le h-\left\lfloor\frac{t}{2}\right\rfloor$ для левых граничных пикселей:

$$a_{I,0}=\begin{cases}a^{+}_{i,j},если \quad \max\{res(\alpha_{i,j})\}\ne\frac{1}{t^2};i=\overline{I-\left\lfloor\frac{t}{2}\right\rfloor,I+\left\lfloor\frac{t}{2}\right\rfloor};j=\overline{1,t};\\ a^{+}_{i,1},если \quad \max\{res(\alpha_{i,j})\}=\frac{1}{t^2};i=\overline{I-\left\lfloor\frac{t}{2}\right\rfloor,I+\left\lfloor\frac{t}{2}\right\rfloor};j=\overline{1,t}.\end{cases}\tag{5}$$

При $I=h+1;\left\lfloor\frac{t}{2}\right\rfloor<J\le w-\left\lfloor\frac{t}{2}\right\rfloor$ для нижних граничных пикселей:

$$a_{h+1,J}=\begin{cases}a^{+}_{i,j},если \quad \max\{res(\alpha_{i,j})\}\ne\frac{1}{t^2};i=\overline{h-t,h};j=\overline{J-\left\lfloor\frac{t}{2}\right\rfloor,J+\left\lfloor\frac{t}{2}\right\rfloor};\\ a^{+}_{h,1},если \quad \max\{res(\alpha_{i,j})\}=\frac{1}{t^2};i=\overline{h+1-t,h};j=\overline{1+t(J-1),Jt}.\end{cases}\tag{6}$$

При $J=w+1;\left\lfloor\frac{t}{2}\right\rfloor<I\le h-\left\lfloor\frac{t}{2}\right\rfloor$ для правых граничных пикселей:

$$a_{I,w+1}=\begin{cases}a^{+}_{i,j},если \quad \max\{res(\alpha_{i,j})\}\ne\frac{1}{t^2};i=\overline{I-\left\lfloor\frac{t}{2}\right\rfloor,I+\left\lfloor\frac{t}{2}\right\rfloor};j=\overline{w-t,w};\\ a^{+}_{i,w},если \quad \max\{res(\alpha_{i,j})\}=\frac{1}{t^2};i=\overline{1+t(I-1),It};j=\overline{w+1-t,w}.\end{cases}\tag{7}$$

При $\left\lfloor \frac{t}{2} \right\rfloor < J \le w - \left\lfloor \frac{t}{2} \right\rfloor; I = 0$ для верхних граничных пикселей:

$$a_{0,J} = \begin{cases} a^{+}_{i,j}, если \quad \max\{res(\alpha_{i,j})\} \ne \frac{1}{t^2}; i = \overline{1,t}; j = \overline{I - \left\lfloor \frac{t}{2} \right\rfloor, I + \left\lfloor \frac{t}{2} \right\rfloor}; \\ a^{+}_{1,j}, если \quad \max\{res(\alpha_{i,j})\} = \frac{1}{t^2}; i = \overline{1,t}; j = \overline{I - \left\lfloor \frac{t}{2} \right\rfloor, I + \left\lfloor \frac{t}{2} \right\rfloor}. \end{cases} \quad (8)$$

Пример. Пусть необходимо выполнить операцию двумерной свертки матриц $A*K$ где, $A = \{a_{i,j}\}; i = 1,h; j = 1,w; h = 5, w = 5;$ центр ядра K равен 1 и для оценки граничных пикселей будут использоваться $t^2 = 3^2$ смежных пикселей.

$$A*K = \begin{pmatrix} 85 & 93 & 236 & 226 & 66 \\ 55 & 13 & 30 & 30 & 194 \\ 185 & 94 & 144 & 46 & 92 \\ 73 & 121 & 207 & 252 & 121 \\ 54 & 219 & 252 & 250 & 9 \end{pmatrix} * \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Для граничных элементов матрицы A алгоритм свертки требуется получить недостающие элементы. Эля этого воспользуемся формулами весового алгоритма (1)-(8). Алгоритм не требует одновременное определение всех «граничных» элементов, а только по мере обращения к ним алгоритма двумерной свертки. Для наглядности определим их все сразу. Не линейное действие алгоритма заметно на подчёркнутых элементах.

$a_{0,0} = a^{+}_{1,1} = 85;$ $a_{6,0} = a^{+}_{5,1} = 54;$ $a_{0,6} = a^{+}_{1,5} = 66;$ $a_{0,0} = a^{+}_{1,1} = 85;$

$a_{1,0} = a^{+}_{1,1} = 85;$ $a_{6,1} = a^{+}_{5,1} = 54;$ $a_{1,6} = a^{+}_{1,5} = 66;$ $a_{0,1} = a^{+}_{1,1} = 85;$

$a_{2,0} = a^{+}_{2,1} = 55;$ $a_{6,2} = a^{+}_{5,2} = 219;$ $\underline{a_{2,6} = a^{+}_{2,3} = a^{+}_{2,4} = 30;}$ $a_{0,2} = a^{+}_{1,2} = 93;$

$a_{3,0} = a^{+}_{3,1} = 185;$ $\underline{a_{6,3} = a^{+}_{4,4} = a^{+}_{5,3} = 252;}$ $\underline{a_{3,6} = a^{+}_{2,3} = a^{+}_{2,4} = 30;}$ $\underline{a_{0,3} = a^{+}_{2,3} = a^{+}_{2,4} = 30;}$

$a_{4,0} = a^{+}_{4,1} = 73;$ $\underline{a_{6,4} = a^{+}_{4,4} = a^{+}_{5,3} = 252;}$ $\underline{a_{4,6} = a^{+}_{4,4} = a^{+}_{5,3} = 252;}$ $\underline{a_{0,4} = a^{+}_{2,3} = a^{+}_{2,4} = 30;}$

$a_{5,0} = a^{+}_{5,1} = 54;$ $a_{6,5} = a^{+}_{5,5} = 9;$ $a_{5,6} = a^{+}_{5,5} = 9;$ $a_{0,5} = a^{+}_{1,5} = 66;$

$a_{6,0} = a^{+}_{5,1} = 54;$ $a_{6,6} = a^{+}_{5,5} = 9;$ $a_{6,6} = a^{+}_{5,5} = 9;$ $a_{0,6} = a^{+}_{1,5} = 66.$

Алгоритм свертки матриц $A*K$ будет работать с граничными элементами доопределенной матрицы $\underline{A^F}$. Матрица $A^F = \{a^F_{I,J}\}; I = \overline{0,h+1}; J = \overline{0,w+1}; h = 5, w = 5;$ состоит только из элементов оригинальной матрицы A и уже вычисленных доопределенных элементов. Матрица A^F указана в примере лишь для наглядности. На практике хранить промежуточную матрицу A^F не требуется, в связи с тем что, не требуется вычислять одновременно все недостающие элементы и для вычислений можно обойтись обращением к элементам оригинальной матрицы A, а не к элементам ее копии.

$$A^F * K = \begin{pmatrix} \underline{85} & \underline{85} & \underline{93} & \underline{30} & \underline{30} & \underline{66} & \underline{66} \\ \underline{85} & 85 & 93 & 236 & 226 & 66 & \underline{66} \\ \underline{55} & 55 & 13 & 30 & 30 & 194 & \underline{30} \\ \underline{185} & 185 & 94 & 144 & 46 & 92 & \underline{30} \\ \underline{73} & 73 & 121 & 207 & 252 & 121 & \underline{252} \\ \underline{54} & 54 & 219 & 252 & 250 & 9 & \underline{9} \\ \underline{54} & \underline{54} & \underline{219} & \underline{252} & \underline{252} & \underline{9} & \underline{9} \end{pmatrix} * \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} \underline{85} & \underline{85} & \underline{93} & \underline{30} & \underline{30} & \underline{66} & \underline{66} \\ \underline{85} & 85 & 93 & 30 & 30 & 66 & \underline{66} \\ \underline{55} & 85 & 93 & 236 & 226 & 66 & \underline{30} \\ \underline{185} & 55 & 13 & 30 & 30 & 194 & \underline{30} \\ \underline{73} & 185 & 94 & 144 & 46 & 92 & \underline{252} \\ \underline{54} & 73 & 121 & 207 & 252 & 121 & \underline{9} \\ \underline{54} & \underline{54} & \underline{219} & \underline{252} & \underline{252} & \underline{9} & \underline{9} \end{pmatrix}.$$

Далее, с учетом размерности оригинальной матрицы A, получим свертку матриц следующего вида:

$$A^F * K \Rightarrow A * K = \begin{pmatrix} \underline{85} & \underline{85} & \underline{93} & \underline{30} & \underline{30} & \underline{66} & \underline{66} \\ \underline{85} & 85 & 93 & 30 & 30 & 66 & \underline{66} \\ \underline{55} & 85 & 93 & 236 & 226 & 66 & \underline{30} \\ \underline{185} & 55 & 13 & 30 & 30 & 194 & \underline{30} \\ \underline{73} & 185 & 94 & 144 & 46 & 92 & \underline{252} \\ \underline{54} & 73 & 121 & 207 & 252 & 121 & \underline{9} \\ \underline{54} & \underline{54} & \underline{219} & \underline{252} & \underline{252} & \underline{9} & \underline{9} \end{pmatrix} \Rightarrow \begin{pmatrix} 85 & 93 & 30 & 30 & 66 \\ 85 & 93 & 236 & 226 & 66 \\ 55 & 13 & 30 & 30 & 194 \\ 185 & 94 & 144 & 46 & 92 \\ 73 & 121 & 207 & 252 & 121 \end{pmatrix}.$$

В результате получена матрица, в которой граничные пиксели были учтены в операции двумерной свертки, а ядро свертки, в случае с граничными пикселями, воздействовало с учетом пикселей вычисленных по весовому принципу. Это значит, что каждый граничный пиксел матрицы получен с учетом близости кода полутона близлежащих пикселей.

Заключение

Проблема необработанных граничных пикселей, возникающая при работе с операциями двумерной свертки для подавления цифрового шума, широко распространена в связи с повсеместным использованием методов на основе свертки в области машинного зрения. Описанный алгоритм позволяет учесть все граничные пиксели при предварительной обработке пиксельного изображения, и решить проблему целого класса методов, а не только отдельных случаев. Особенность представленного алгоритма в том, что он не предполагает хранение промежуточного изображения, а лишь обращения к элементам оригинальной матрицы. Это экономит ресурсы системы и одновременно заполняет недостающие пиксели близкими по полутону значениям по весовому критерию. Алгоритм позволяет варьировать величину t , отвечающую за рассмотрение t^2 близлежащих элементов пиксельной матрицы для граничного пикселя и тем самым подстраиваться под вычислительные возможности используемой системы. Отсутствие жестких ограничений позволяет использовать алгоритм для широкого спектра методов, где задействована операция двумерной свертки.

ЛИТЕРАТУРА

1. **Гашников М. В.** Методы компьютерной обработки изображений., Методы компьютерной обработки изображений / Под ред. В. А. Сойфера. – 2-е изд., испр. – М.: ФИЗМАТЛИТ, 2003. – 784 с.
2. **Яне Б.** Цифровая обработка изображений. Москва: Техносфера, 2007. – 584с.
3. **Фисенко В. Т., Фисенко Т. Ю.** Компьютерная обработка и распознавание изображений: учеб. пособие. – СПб: СПбГУ ИТМО, 2008. –192 с.
4. **Форсайт Д., Понс Ж.** Компьютерное зрение. Современный подход. – Вильямс, 2004. – 928 с.
5. **Шапиро Л.** Компьютерное зрение / Л Шапиро, Дж. Стокман; Пер. с анг. – М.: БИНОМ. Лаборатория знаний, 2006. – 752 с.
6. **Bailey D. G.** Image border management for FPGA based filters. In: 6th IEEE international symposium on electronic design, test and applications, Queenstown, 17–19 Jan 2011, pp 144–149.
7. **Брейсуэлл Р. Н.** Преобразование Хартли: Пер с англ. – М.: Мир, 1990. – 175 с.
8. **Хиршман И. И., Уиддер Д. В.** Преобразования типа свертки. М.: Издательство иностранной литературы, 1958. – 312 с.
9. **Оппенгейм А., Шафер Р.** Цифровая обработка сигналов, Москва, Связь, 1979.
10. **Заерко Д. В.** Весовой метод решения проблемы граничных пикселей в алгоритме сверточной фильтрации цифрового шума / Д. В. Заерко, В. А. Липницкий // Кодирование и цифровая обработка сигналов в инфокоммуникациях: материалы международной научно-практической конференции, Минск, 24 апреля 2020 г. /; редкол.: В. К. Конопелько, В. Ю. Цветков, Л. А. Шичко. – Минск, 2020. – С. 59–63.

REFERENCES

1. **M. V. Gashnikov, N. I. Glumov, N. Yu. Il'yasova, V. V. Myasnikov, et al.,** Computer Image Processing Methods, Ed. by V. A. Soifer, Fizmatlit, Moscow, 2003. – 784 p.
2. **Yane B.** Cifrovaya obrabotka izobrazhenij. Moskva: Tekhnosfera, 2007. – 584p.
3. **Fisenko V. T., Fisenko T. U.** Komp'yuternaya obrabotka i raspoznavanie izobrazhenij: ucheb. posobie. – SPb: SPbGU ITMO, 2008. –192 p.
4. **Forsyth. D, Ponce. J.** Computer vision: a modern approach. Upper Saddle River, N.J., Prentice Hall, 2003.
5. **Shapiro L.** Kompyuternoezrenie / L Shapiro, Dzh. Stokman; Per. s ang. – M.: BINOM. Laboratoriy aznanij, 2006. – 752 p.
6. **Bailey D. G.** Image border management for FPGA based filters. In: 6th IEEE international symposium on electronic design, test and applications, Queenstown, 17–19 Jan 2011, pp 144–149.
7. **Ronald N. B.** The Hartley Transform. //Oxford University Press, Inc. 1986 Madison Ave. New York, NY United States.
8. **Hirshman I. I., Uidder D. V.** Convolution / Hirshman I. I, Uidder D. V. Conversion type svertki, 1958. – 312 p.
9. **Oppenheim A., Schafer R.** Digital signal processing, 1979. – 283 p.
10. **Zaerko, D. V.** Weighted method solving of boundary pixels problem in digital noise convolution filtering algorithm / D. V. Zaerko, V. A. Lipnitski // Kodirovanie i cifrovaya obrabotka signalov infokommunikaciyah: materials of the international scientific conference, BSUIR, Minsk, Rep. Belarus, 24 April 2020 г. – P. 59–63.

Поступила	*После доработки*	*Принята к печати*
01.08.2020	*01.12.2020*	*01.12.2020*

ZAERKO D. V., LIPNITSKI V. A.

WEIGHTED DETERMINATION ALGORITM OF BOUNDARY PIXELS

Belarusian State University of Informatics and Radioelectronics, Minks, Republic Belarus

While working with digital noise reduction techniques, which are based on theory of convolution matrix and used convolution operation, it necessary to use algorithms to bypass boundary pixels in the image pixel matrix. The problem exists because convolution itself algorithm have peculiarity, it mean that peculiarity convolution kernel used to each element of pixel matrix. That feature characterize a lot of classes of methods which used idea of convolution matrix. There are a lot of primitive ways to solve it, but none of these ways made a consensus between economical use of resources and filling border pixels with colour coding, which is not so far from colours of corresponding pixels. The object of research is pixel matrix of image. The subject of study is algorithms for filling boundary pixels when superimposing a convolution matrix on a pixel matrix of an image. The main target is creating of effective filled algorithm for border pixels which are close to code colour to relation pixels for used in convolution matrix. Filled border pixels will use to operation convolution for each pixels original image. Algorithm of filled border pixels by step of applied convolution kernel anchors to the pixel, when pixel accessing in convolution algorithm goes beyond the pixel matrix of the original image. Algorithm takes into account the «special» cases of overstepping and allows to do fast calculation to determine the colour code of the missing pixel. The algorithm is simple to program and easily integrates with the basic convolution matrix algorithm in digital image defects.

Keywords: *pixel matrix, digital image defects, digital noise suppression, 2D convolution, noise filtering, boundary pixels, halftone image.*

Заерко Денис Владимирович, аспирант кафедры информатики БГУИР, область научных исследований – машинные алгоритмы распознавания объектов, машинное обучение, анализ данных.
Zaerko. D., postgraduate student, Informatics department of BSUR.
Email: zaerko1991@gmail.com

Липницкий Валерий Антонович, профессор, доктор технических наук, кафедра информатики БГУИР, область научных интересов – алгебра и ее приложения, защита информации от помех и несанкционированного доступа.

Lipnitski.V.A., Doctor of technical sciences, Professor, Informatics department of BSUR.

E-mail: valipnitski@yandex.by.