НИЯУ МИФИ, факультет Кибернетики, Каф. 17

Компьютерная обработка изображений

Лекция 8: Морфологические операции.

Сафонов И.В., Крыжановский К.А., Егорова М.А.

2011

1

Морфологические операции

Морфоло́гия (от греческого μορφή «форма» и λογία «наука») в широком понимании — наука о формах и строении.

Математическая морфология (*mathematical morphology*) – теория и набор алгоритмов для анализа геометрических структур, основанная на теории множеств.

В обработке изображений для существует ряд морфологических операций (фильтров), основанных на аппарате математической морфологии и предназначенных для обработки бинарных изображений.

Большинство морфологических фильтров легко обобщаются для обработки полутоновых изображений.

Существуют также несколько подходов для обобщения морфологических операций для цветных изображений, но единого подхода пока нет, и в данной области продолжаются исследования.

2

Основные понятия теории множеств

Объединение: $C = A \cup B$

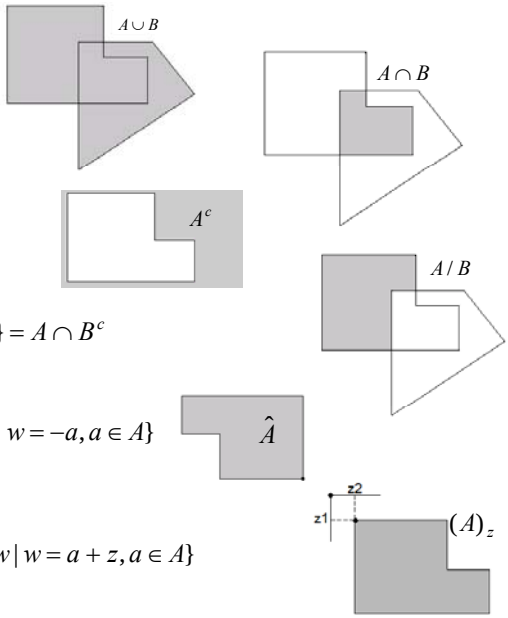
Пересечение: $D = A \cap B$

Дополнение: $A^c = \{w \mid w \notin A\}$

Разность: $A / B = \{w \mid w \in A, w \notin B\} = A \cap B^c$

Центральное отражение: $\hat{A} = \{w \mid w = -a, a \in A\}$

Параллельный перенос: $(A)_z = \{w \mid w = a + z, a \in A\}$



Основные морфологические операции

Эрозия: $A \Theta B = \{z \mid (B)_z \subseteq A\}$

Наращение: $A \oplus B = \{z \mid (\hat{B})_z \cap A = \emptyset\}$

Открытие: $A \circ B = (A \Theta B) \oplus B$

Закрытие: $A \bullet B = (A \oplus B) \Theta B$

Верх шляпы: $th = A - (A \circ B)$

Низ шляпы: $bh = (A \bullet B) - A$

Морфологический градиент: $g = (A \oplus B) - (A \Theta B)$

Преобразование успех/неудача: $A \otimes B = (A \Theta B_1) \cap (A^c \Theta B_2)$

B – структурный элемент/маска/апертура:

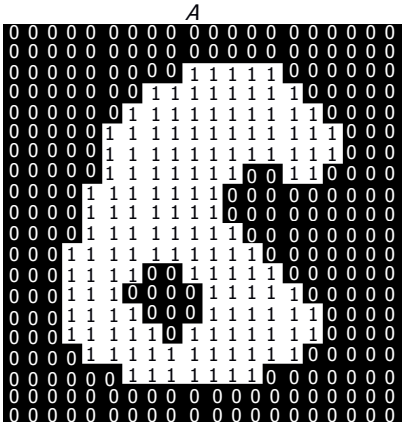
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Эрозия (*Erosion*)

$$A \Theta B = \{z \mid (B)_z \subseteq A\}$$

Если текущий пиксел - пиксел объекта и хотя бы один из пикселов изображения соответствующих ненулевому элементу маски является пикселом фона, то значение текущего пиксела изменяется и становится пикселом фона.

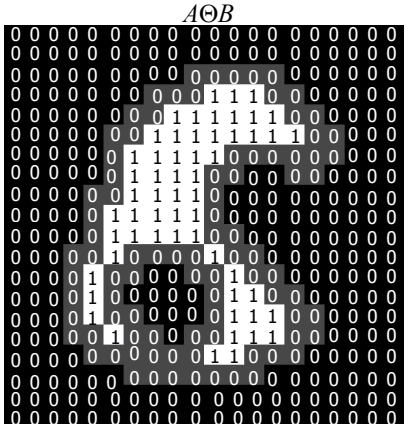
A



B

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$A \Theta B$

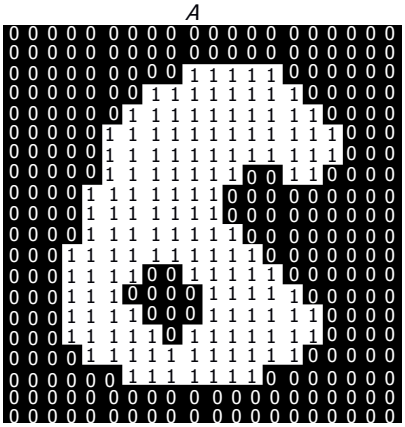


Наращение (*Dilation*)

$$A \oplus B = \{z \mid (\hat{B})_z \cap A = \emptyset\}$$

Если текущий пиксел - пиксел фона и хотя бы один из пикселов изображения соответствующих ненулевому элементу маски является пикселом объекта, то значение текущего пиксела изменяется и становится пикселом объекта.

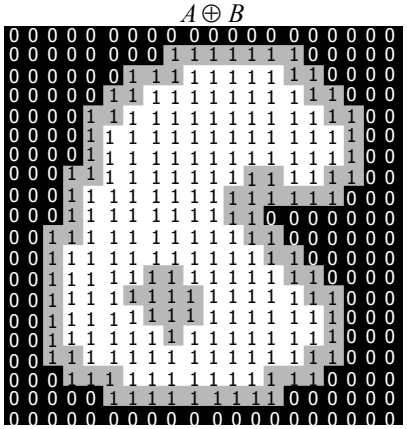
A



B

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$A \oplus B$

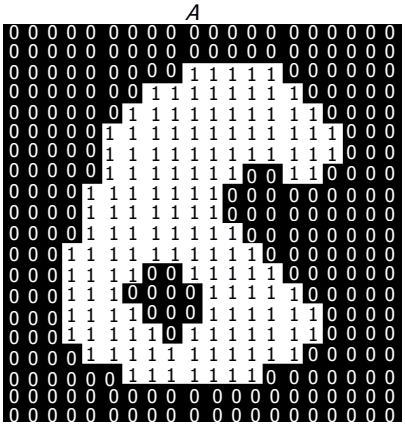


Открытие/Размыкание (*Open*)

$$A \circ B = (A \Theta B) \oplus B$$

Открытие есть последовательное применение эрозии и наращения с одинаковым структурным элементом. Открытие позволяет открыть «дыры» объекта, находящиеся около границы, а также удалить незначительные неровности на границе объекта.

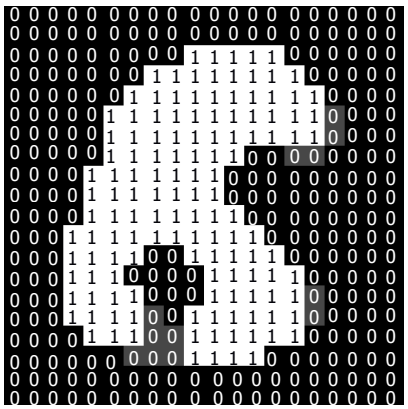
A



B

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$A \circ B$



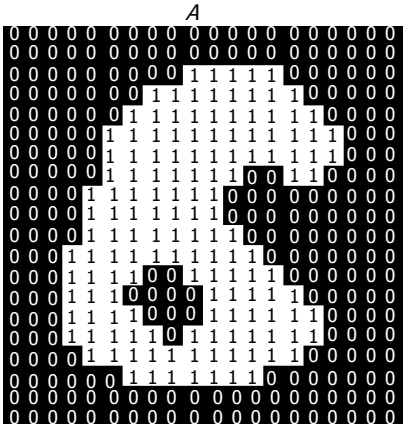
7

Закрытие/Замыкание (*Close*)

$$A \bullet B = (A \oplus B) \Theta B$$

Закрытие есть последовательное применение наращения и эрозии с одинаковым структурным элементом. В результате получается объект того же размера, но без «дырок» и впадин на границе.

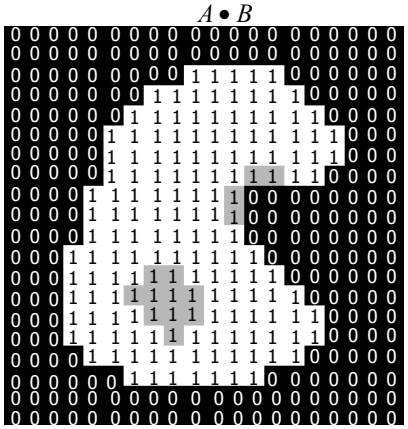
A



B

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$A \bullet B$

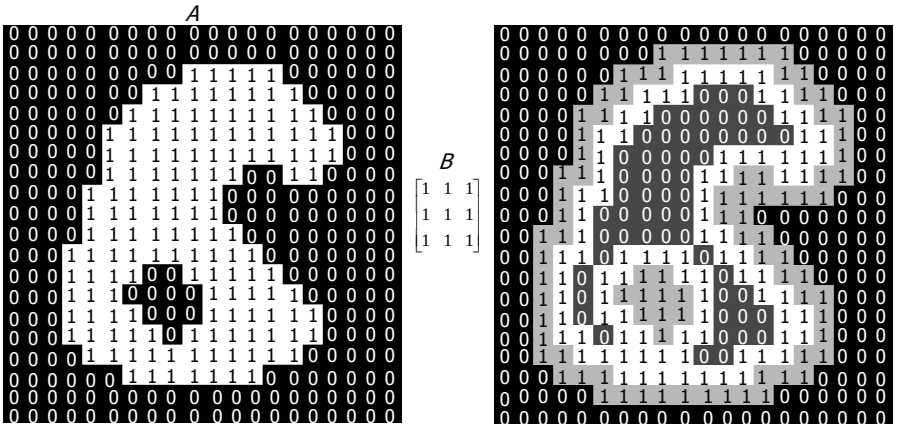


8

Морфологический градиент

$$g = (A \oplus B) - (A \Theta B)$$

Морфологический градиент есть разность изображений полученных в результате наращения и эрозии с одинаковым структурным элементом. Применение этой операции позволяет получить границу объекта.

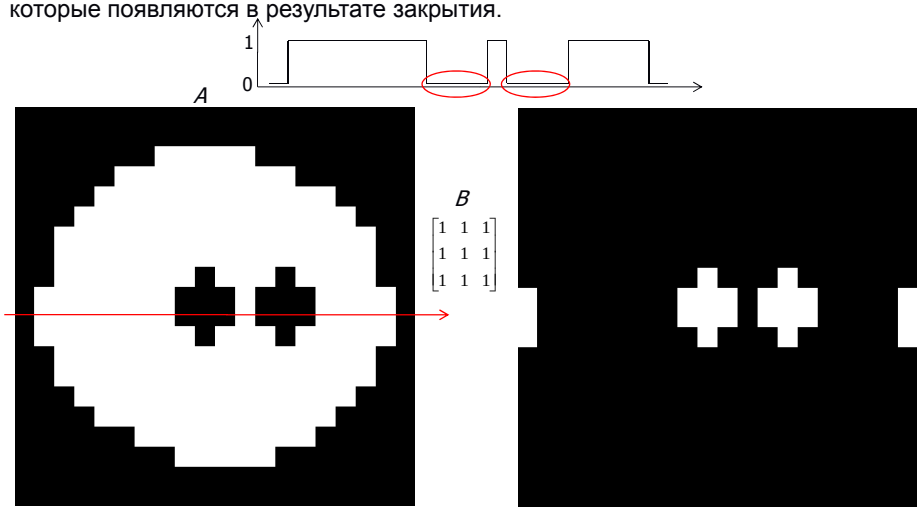


9

Низ шляпы (*Bothat*)

$$bh = (A \bullet B) - A$$

Bothat - есть результат разности закрытия исходного изображения и самого исходного изображения. Т.е. результат содержит те пикселы изображения, которые появляются в результате закрытия.

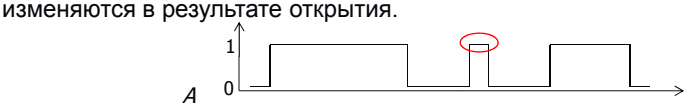


10

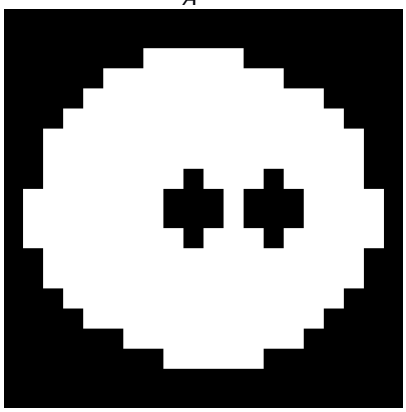
Верх шляпы (*Tophat*)

$$th = A - (A \circ B)$$

Tophat - есть результат разности исходного изображения и открытия исходного изображения. Т.е. результат содержит те пикселы изображения, которые изменяются в результате открытия.

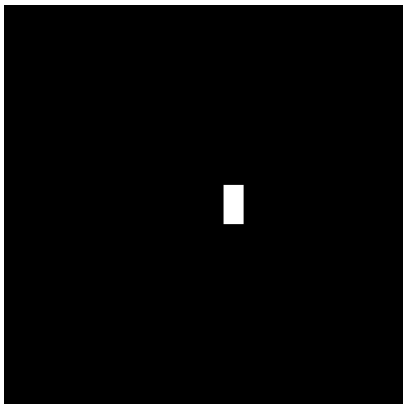


A



B

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



Преобразование успех/неудача (*Hit/miss*)

$$A \otimes B = (A \Theta B_1) \cap (A^c \Theta B_2)$$

В преобразовании используется составной структурный элемент B=(B_1, B_2). Цель операции найти пикселы x, для которых B_1 содержится в A, а B_2 в A^c

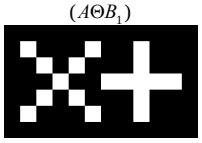
Данное преобразование используется для выделения на изображении объектов определенных размеров и формы, задаваемой структурным элементом.

Найти пикселы объекта, у которых нет 4-х связных соседей.

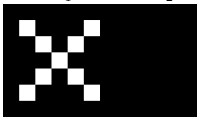
A

$B1$

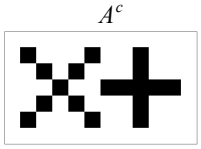
$(A \Theta B_1)$



$(A \Theta B_1) \cap (A^c \Theta B_2)$

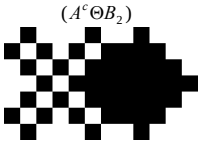


A^c



$B2$

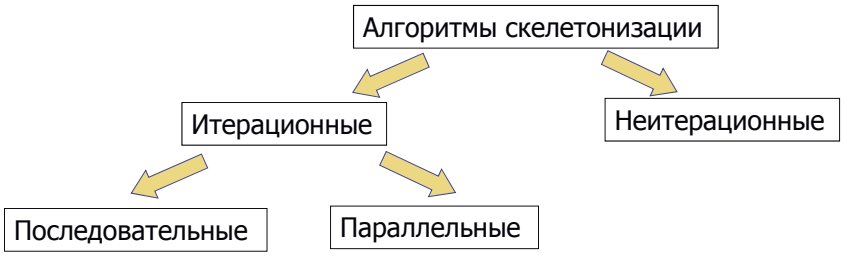
$(A^c \Theta B_2)$



Пример взят из презентации Theo Schouten

Построение остова

Остов (скелет, скелетон) – некоторая серединная линия объекта. Построение остова – утоньшение (*thinning*) объекта до связанной области толщиной в один пиксел.



Неитерационные алгоритмы основаны на эвристической обработке объектов определенной формы. В результате их применения получается центральные (медианные) линии объектов за одну итерацию без анализа каждого отдельного пикселя.

13

Итерационные алгоритмы утоньшения

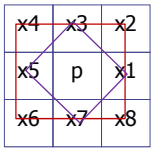
Итерационные алгоритмы удаляют слой за слоем пиксели на границе объекта до тех пор, пока не останется только связная область толщиной в один пиксел. На каждой итерации растр сканируется слева направо и сверху вниз и для каждого пикселя решается, подлежит он удалению или нет.

В последовательных алгоритмах удаление пикселя *p* на n-ной итерации зависит от результатов (*n-1*) –ой итерации, а так же от пикселей, уже обработанных на этой итерации.
В параллельных алгоритмах пиксели исследуются на удаление на основе результатов, полученных только на предыдущей итерации.

Далее более подробно рассмотрены:
- алгоритм последовательного утоньшения Хилдича (*Hilditch*)
- алгоритм параллельного утоньшения Рутовица (*Rutovitz*)
- алгоритм параллельного утоньшения Ахмеда-Варда (*Ahmed*, *Ward*)

14

Число пересечений (*Crossing number*)



Число пересечений используется для оценки связности.

b(p) – количество пикселов в окрестности p, относящихся к объекту.

$$b(p) = \sum_{i=1}^{8} x_i$$

$X_R(p)$ – число пересечений Рутовица:

$$X_R(p) = \sum_{i=1}^{8} |x_{i+1} - x_i|, \quad \text{если } (i+1)>8, \text{ то } i=i-8.$$

Физический смысл: Количество переходов с фона на объект и наоборот при обходе вокруг пиксела *p*

$X_H(p)$ – число пересечений Хилдича:

$$X_H(p) = \sum_{i=1}^{4} k_i, \qquad k_i = \begin{cases} 1, & x_{2i-1} = 0 \quad и \quad (x_{2i} = 1 \quad или \quad x_{2i+1} = 1) \\ 0, & иначе \end{cases}$$

Физический смысл: Количество переходов с фона на объект при обходе по 4-м соседям пиксела *p*

15

Алгоритм Хилдича (1)

Удаляемый пиксел должен удовлетворять следующим условиям:

Условия общие для всех алгоритмов утоньшения

1. p=1 (пиксел принадлежит объекту)
2. b(p)≥2 (пиксел не изолирован, т.е. после удаления данного пиксела у него останутся соседи)
3. b(p)<8 (p является контурным или граничным пикселом, т.е. имеет хотя бы одного из 4-х соседей относящегося к фону)

H1. Как минимум один из 8-ми соседей *p* должен быть не помечен на удаление
Предотвращает чрезмерную эрозию малых циклических последовательностей и предотвращает удаление одиночных пикселей

H2. $X_H(p) = 1$
Поддерживает связанность

H3. При помеченном на удаление x_3 установка $x_3 = 0$ не изменит числа пересечений $X_H(p)$
H4. При помеченном на удаление x_5 установка $x_5 = 0$ не изменит числа пересечений $X_H(p)$
Предотвращает удаление линий толщиной 2 пикселя

16

Алгоритм Хилдича (2)

17

Алгоритм Рутовица (1)

Удаляемый пиксел должен удовлетворять следующим условиям:

Условия общие для всех алгоритмов утоньшения

1. p=1 (пиксел принадлежит объекту)
2. b(p)≥2 (пиксел не изолирован, т.е. после удаления данного пиксела у него останутся соседи)
3. b(p)<8 (p является контурным или граничным пикселом, т.е. имеет хотя бы одного из 4-х соседей относящегося к фону)

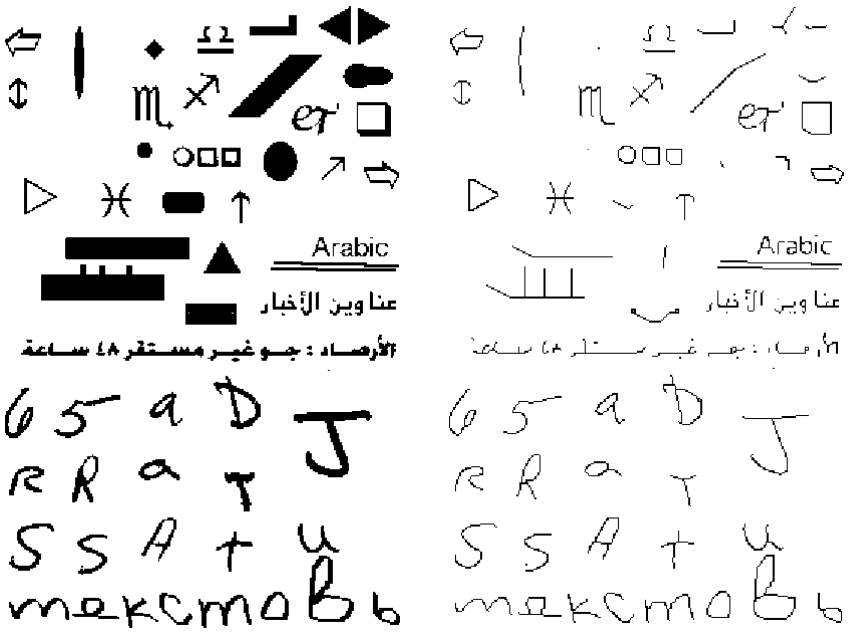
R1. $X_R(p) = 2$

R2. $x_1x_3x_5 = 0$ или $X_R(x_3) \neq 2$

R3. $x_7x_1x_3 = 0$ или $X_R(x_1) \neq 2$

18

Алгоритм Рутовица (2)



19

Реализация алгоритма Рутовица (1)

Пример реализации алгоритма Рутовица для бинарного изображения, которое развернуто в 8 bpp.

```
// Структура для хранения рабочих данных алгоритма
struct THINNINGCONTEXT {
    int             width;      // Ширина изображения
    int             height;     // Высота изображения
    unsigned char*  image;      // Рабочее изображение
    bool*           deleted;    // Буфер разметки удаляемых пикселов
};
// Сборка соседних пикселов в отдельный массив
void GatherNeighbors(int x, int y, unsigned char nbr[9], THINNINGCONTEXT& c)
{ // Индексы соседей:
  // 7 0/8 1
  // 6  *  2
  // 5  4  3
// Смещения соседей относительно текущего пиксела
const int shift[9][2]={{0,-1},{1,-1},{1,0},{1,1},{0,1},{-1,1},{-1,0},{-1,-1},{0,-1}};
    for( int i = 0; i < 9; ++i )
    {
        int nx = x + shift[i][0],
            ny = y + shift[i][1];
        if( nx >= 0 && nx < c.width &&
            ny >= 0 && ny < c.height )
            nbr[i] = c.image[ny * c.width + nx];
        else
            nbr[i] = 0;
    }
}
```

20

Реализация алгоритма Рутовица (2)

```
// Вычисление числа Рутовица
int RutovitzNumber( const unsigned char nbr[9] )
{
    int num = 0;
    for( int i = 0; i < 8; ++i )
        num += abs((int)nbr[i+1] - (int)nbr[i]);
    return num;
}

// Выполняет итерацию утоньшения. Возвращает true в том случае, если
// есть пикселы - кандидаты на удаление.
bool ThinRutovitzIteration(int x, int y, THINNINGCONTEXT& c) {
    unsigned char nbr[9];                  // соседи пиксела
    GatherNeighbors(x, y, nbr, c);
    int Nr = RutovitzNumber(nbr);
    int r1 = nbr[0] * nbr[2] * nbr[4];     //условия 3 и 4;
    int r2 = nbr[0] * nbr[2] * nbr[6];
    int b = 0;            // проверка, что пиксел не изолированный
    for( int k = 0; k < 8; ++k )
        b += nbr[k];
    if( (Nr == 2) && (b > 1) && (b < 7) && (r1 == 0) && (r2 == 0) )
    {
        c.deleted[y * c.width + x] = true;
        return true;
    }
    return false;
}
```

21

Реализация алгоритма Рутовица (3)

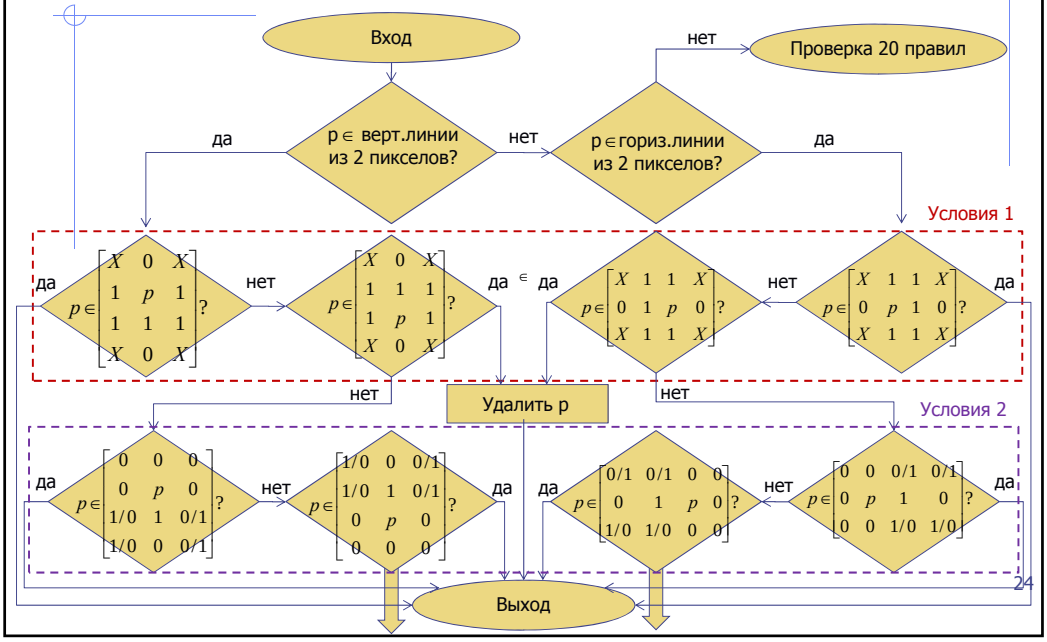
```
// Обнуляет все отмеченные пикселы
void DeletePixels( THINNINGCONTEXT& c )
{
    const bool*    pd = c.deleted;
    unsigned char* pi = c.image;
    for( int y = 0; y < c.height; ++y )
    {
        for( int x = 0; x < c.width; ++x )
        {
            if( pd[x] ) pi[x] = 0;
        }
        pd += c.width;
        pi += c.width;
    }
}

void ThinRutovitz( const unsigned char* src, unsigned char* dst,
                   int width, int height )
{
    THINNINGCONTEXT c;
    c.width  = width;
    c.height = height;
    c.image  = dst;
    int pixelCount = width * height;
// Выделяем и инициализируем буффер для разметки удаляемых пикселов
...
```

22

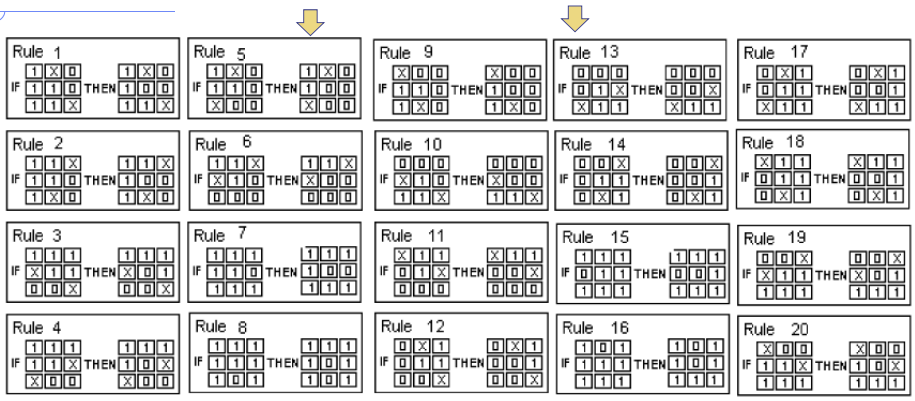
Реализация алгоритма Рутовица (4)

```
...
// Выделяем и инициализируем буффер для разметки удаляемых пикселов
    c.deleted = new bool[pixelCount];
    for( int t = 0; t < pixelCount; ++t ) c.deleted[t] = false;
    // Копируем исходное изображение в рабочий буффер
    for( int t = 0; t < pixelCount; ++t ) c.image[t] = src[t];
    // Запускаем алгоритм Рутовица
    bool isChanged = true;
    while( isChanged )
    {
        isChanged = false;
        for( int y = 0; y < height; ++y )
        {
            for( int x = 0; x < width; ++x )
            {
                if( c.image[y * width + x] == 1 )
                {
                    if( ThinRutovitzIteration(x, y, c) )
                        isChanged = true;
                }
            }
        }
        DeletePixels(c);
    }
    // Освобождаем буфер разметки
    delete[] c.deleted;
}
```


Алгоритм Ахмеда-Варда (1)



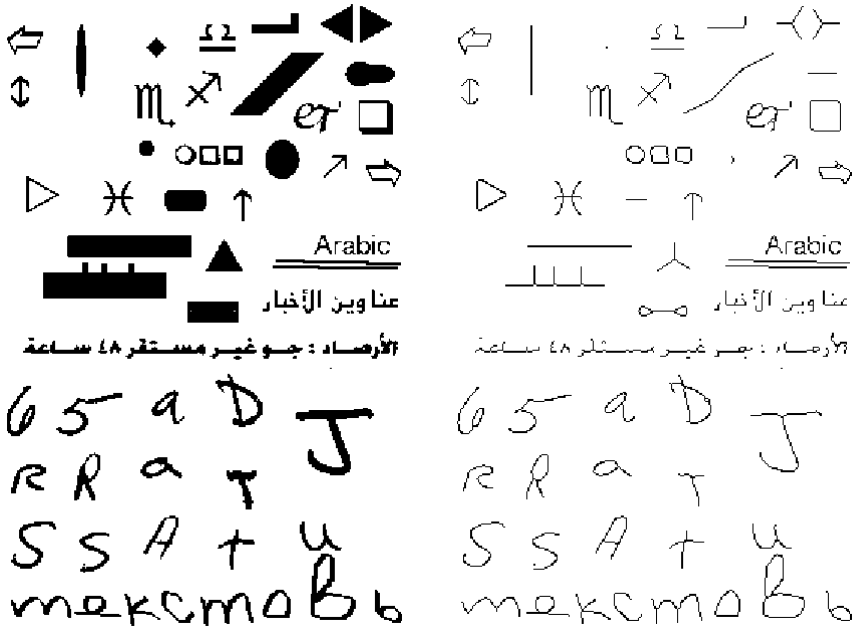
Алгоритм Ахмеда-Варда (2)



Смысл условий 1: Предотвращение удаления линий толщиной 2 пиксела
Смысл условий 2: Предотвращение удаления зигзагообразной диагональной линии.
Смысл 20 правил: Пикселы считаются граничными и подлежат удалению в том случае, если 8 соседей исследуемого пикселя образуют две и только две последовательности черного и белого цвета, причем пикселей черного цвета должно быть не меньше двух.

Запись $\begin{bmatrix} 0 & 0 & 0 \\ 0 & p & 0 \\ 1/0 & 1 & 0/1 \\ 1/0 & 0 & 0/1 \end{bmatrix}$ следует читать как: $\begin{bmatrix} 0 & 0 & 0 \\ 0 & p & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$ или $\begin{bmatrix} 0 & 0 & 0 \\ 0 & p & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$

Алгоритм Ахмеда-Варда (3)



Предельная эрозия (*Ultimate erosion*)

Предельная эрозия применяется для нахождения количества выпуклых объектов (или выпуклых частей для невыпуклых объектов) на изображении.

Операция эрозии применяется к объекту до тех пор пока объект полностью не исчезнет, тогда в результирующее изображение переносят часть объекта с предыдущей операции эрозии. И так для всех объектов на изображении.

Исходное изображение

Результат применения эрозии

Результат предельной эрозии

Объекты, исчезающие на следующем шаге эрозии

27

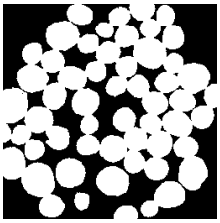
Пример взят из материалов лекции Maintz

Пример использования морфологических операций

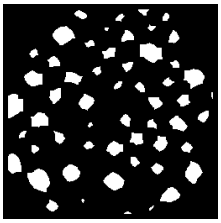
Задача: разъединить слипшиеся объекты округлой формы с помощью морфологических операций.

1. Эрозия до тех пор, пока не исчезнет слипание объектов.
2. Построение остова фона
3. Умножение исходного бинарного изображения и остова фона

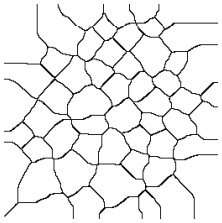
Исходное изображение



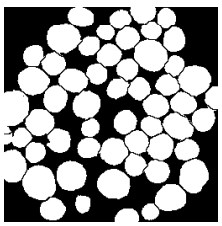
Результат нескольких итераций эрозии



Остов фона



Изображение с разделенными объектами



28

Обобщение на полутоновые изображения (1)

Полутоновая дилатация изображения *f* по примитиву *b*:

$$(f \oplus b)_{(x,y)} = \max\{f(x-x', y-y') + b(x', y') \mid (x', y') \in D_b\}$$

где D_b – двоичная матрица, задающая область определения *b*

Полутоновая эрозия изображения *f* по примитиву *b*:

$$(f \ominus b)_{(x,y)} = \min\{f(x+x', y+y') - b(x', y') \mid (x', y') \in D_b\}$$

В случае плоского примитива (значение которого на всей области определения одинаково) операция эрозии есть применение к изображению минимального фильтра, операция наращения – применение максимального фильтра..

$$(f \oplus b)_{(x,y)} = \max\{f(x-x', y-y') \mid (x', y') \in D_b\}$$
$$(f \ominus b)_{(x,y)} = \min\{f(x+x', y+y') \mid (x', y') \in D_b\}$$

29

Обобщение на полутоновые изображения (2)

Исходное изображение

Наращение

Эрозия

Закрытие

Открытие

30