ПРЕДВАРИТЕЛНА ОБРАБОТКА НА ИЗОБРАЖЕНИЯ СЪС СИМВОЛНА ИНФОРМАЦИЯ

Гочо Гочев, Цветан Шалявски, Иван Трифонов

Резюме: *Разглежда се силно зашумено полутоново изображение с цифрова информация. Символите не са центрирани. Задачата е да се отделят цифрите, да се отхвърли шумът и да се центрират символите в конкретно входно поле. Извършва се филтрация на входното поле. Адаптивно се определя праг за бинаризация на полутоновото изображение. За полученото двоично изображение се използуват евристични алгоритми за подобряване качеството на изображението. Всяка цифра се центрира спрямо геометричния център на избраното входно поле за символите. Центрираните изображения на символите се подават към входа на невронна мрежа за тяхното разпознаване.*

PROCESSING OF IMAGES WITH SYMBOL INFORMATION

Gotcho Gotchev, Tzvetan Shaliavsky, Ivan Trifonov

Abstract: *A gray-scaled input image with high degree of noise is being processed. Symbols are not centered. The task is to separate the digits; to filter the noise and after that to center the symbols in a particular input field. Noise reduction is made. A binary threshold value is calculated. Heuristic algorithms are used for improving the quality of the resulting binary image. The digits are centered by its geometrical center. The result images are fed to the input of a neural network for recognition.*

1. УВОД

В статията предварителната обработка на изображението се разглежда в рамките на подготовка за разпознаване на символна информация, което определя появата на някои специфични операции и отпадането на други (като компресия например). Тя е важен етап от разпознаването на образи, като основната му цел е да намали случайните шумове при формирането на функцията на изображението $f(x,y)$, да центрира символите във входното поле.

Представянето на $f(x,y)$ се свежда до представяне на дискретната функция $g(x,y)$ в паметта като числов масив: $g(i,j)$, $i = 0,1,...,N-1$; $j = 0,1,...,M-1$, където $N \times M$ определя броя на пикселите на входното изображение. Ако за всеки пиксел се определят 256 нива на сиво, то необходимата памет за едно входно изображение е $N \times M$ байта. За цветни изображения тази памет нараства значително.

Годишник на ТУ–София, т.48, кн.4/1995г.
Proceedings of the Technical University – Sofia, v.48, b.4/1995.

2. АЛГОРИТМИ ЗА ПРЕДВАРИТЕЛНА ОБРАБОТКА НА ИЗОБРАЖЕНИЯТА

2.1. "Изостряне" (sharpen) "омекотяване" (soften) на образи

Преди "изостряне" или "омекотяване" се извършва филтрация на полутоновото визуално изображение с цел премахването на случайни шумове. Реализирани са филтър по средна стойност и медианен филтър, чиито описания са дадени в [1].

За "изостряне" на изображенията се използува следната преобразуваща матрица:

$$A = \begin{vmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{vmatrix}.$$

Коефициентите могат да се избират произволно, като единственото условие е тяхната сума да е 1. При други коефициенти ефектът на изостряне ще бъде засилен или смекчен в зависимост от коефициентите.

Например матрицата A_1:

$$A_1 = \begin{vmatrix} 0 & -2 & 0 \\ -2 & 9 & -2 \\ 0 & -2 & 0 \end{vmatrix}$$

служи за "изостряне" на образи, като ефектът на "изостряне" е поголям.

Важно е да се отбележи, че при "изостряне" не се получават нови изображения, а всъщност това е изходното изображение, в което се виждат повече детайли, т.е. появяват се детайли, които са били скрити за зрението ни. Ефектът на "изостряне" е особено полезен за обработка на полутонови изображения, които след това ще се бинеаризират. Не се препоръчва "изостряне" на изображенията, ако в изходното полутоново изображение действителният брой на нивата на квантуване е малък (например по-тъмни изображения). Неколкратното "изостряне" на изображението води до влошаване на качеството му. Препоръчва се еднократно изостряне, а ако ефектът е недостатъчен, се сменя изострящата матрица. Изострянето на изображенията се реализира с функцията *ispSharpen*.

За "омекотяване" се използува подобна матрица, с тази разлика, че коефициентите са реални числа между [0,1]. Естествено за ускорение

може да се използува целочислена аритметика и числата да се умножат по определена константа и след това да се разделят на нея (например 100). Отново изискване за коефициентите е сумата им да дава единица.

Ето една примерна матрица, използувана във функцията *ispSharpen:*

$$A_2 = \begin{vmatrix} 0.07 & 0.12 & 0.07 \\ 0.12 & 0.28 & 0.12 \\ 0.07 & 0.12 & 0.07 \end{vmatrix}$$

Като цяло тези филтри могат да бъдат дефинирани и матрично по следният начин:

$$g'(i,j) = F\left(\mathbf{A}*\mathbf{G}\langle i,j\rangle\right),$$

A е матрицата 3×3 на преобразуването; $\mathbf{G}\langle i,j\rangle$ е матрицата с център (i,j); $F(A)$ е функция, която връща като резултат сумата от коефициентите на входния ѝ параметър - матрицата **A**;

Ето и обобщен алгоритъм за работа:

1) нулира се изходният масив;
2) за $i = 1,2,...,N-2$;
3) за $j = 1,2,...,M-2$;
4) $g'(i,j) = F\left(\mathbf{A}*\mathbf{G}\langle i,j\rangle\right)$,
5) край.

2.2. Определяне на прага за бинаризация на изображението

За определяне на прага на бинаризация се използува модифициран вариант на хистограма - средноаритметично с тегла [2].

$$\bar{x}' = \frac{x_1h_1 + x_2h_2 + \ldots + x_nh_n}{h_1 + h_2 + \ldots + h_n} = \frac{[hx]}{[h]} = \frac{1}{k}\sum_{i=1}^{n} x_ih_i,$$

където $k = \sum_{i=1}^{n} h_i$; h_i е номерът на съответния цвят; x_i е брой появявания за съответния цвят.

Според теоремата на Коши

$$x_{\max} > x_{cp.kb.} > \bar{x} > x_{cp.x} < x_{\min}.$$

Това гарантира, че средната стойност, която се получава по формулата, се намира в интервала $[X_{\min}, X_{\max}]$. Получената по тази формула

стойност за бинарния праг е малко по-малка от необходимата. Затова се коригира с около 20-22% от стойността ѝ.

Алгоритъмът на функцията за определяне на бинарния праг е следният:

алгоритъм 1:

1) за всеки цвят се преброява колко пъти се среща в изображението. Запазва се резултатът в масив p с брой елементи равен на броя на цветовете.

2) полага се $a_1 = b_1 = 0$;

3) за $i = 0,1,...,$ *max_брой_цветове*-1 се изчислява:

$a_1 += (i+1)*p[i]$;

$b_1 += p[i]$;

4) бинарният праг b е равен на

$b = (a_1/b_1)*$ ***ADJUST_VALUE;***

5) край,

където $p[i]$ е честотата на появяването на i-тия цвят; ***ADJUST_VALUE*** е коефициент за донастройване на прага. ***ADJUST_VALUE*** е евристично определяна константа и зависи от физическите характеристики на сканиращата апаратура.

Алгоритъмът на функцията за бинаризиране на изображението до две нива - 1 бяла точка, 0 - черна точка е:

алгоритъм 2:

1) определя се бинарният праг b по ***алгоритъм 1;***

2) за $i = 1,2,...,N-1$;

3) за $j = 1,2,...,M-1$;

4) ако $g[i][j] < b$, то $g[i][j] = 0$, иначе $g[i][j] = 1$

5) край.

Полученото изображение вече е с две нива на квантуване и може да се използува за разпознаване на символи, чиито алгоритми по принцип използуват двоичните изображения. Означението за бял и черен цвят е условно и може да се променя. Отделянето на границите за обектите в изображението може да се извърши, използувайки диференциални оператори. Най-често се използува операторът на Собел с апертура 3×3 [2].

2.3. Двоична филтрация

Двоичната филтрация се прилага за двоични изображения, като "поправя" следните видове смущения: празнини и прекъсвания, изолирани петна, диагонални смущения, лъжливи крайни точки.

Реализацията на алгоритъм за двоична филтрация налага определянето на следните понятия [3]:

- дефинира се множество атоми $\mathbf{A} = \{\mathbf{A}_1, \mathbf{A}_0\}$, където $\mathbf{A}_1$ е подмножество от всички атоми (пиксели), принадлежащи на контурите на обектите, а A_0 е подмножеството от атоми, които принадлежат на фона. От дефинициията следват: $\mathbf{A}_1 \cup \mathbf{A}_0 = \mathbf{A}$; $\mathbf{A}_1 \cap \mathbf{A}_0 = 0$. Дефинира се всеки атом с двойката $g(i, j)$ от функцията на входното изображение.
- Счита се, че всеки атом е свързан със съседните му атоми в апертура 3×3, чиито център е самият атом $(a_{ij} = g(i, j))$.
- полага се:

$$x_0 = g(i-1, j-1), \quad x_1 = g(i-1, j), \quad x_2 = g(i-1, j+1),$$

$$x_3 = g(i, j-1), \quad x_4 = g(i, j+1),$$

$$x_5 = g(i+1, j-1), \quad x_6 = g(i+1, j), \quad x_7 = g(i+1, j+1),$$

$$\mathbf{V} = \{x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7\};$$

По тези формули се дефинират следните преобразуващи правила:

$$S_1: g'(i, j) = \left((x_0 \cup x_1 \cup x_2)(x_4 \cup x_5 \cup x_6)(x_2 \cup x_3 \cup x_4)(x_0 \cup x_7 \cup x_6)\right) \bar{g}(i, j)$$

$$S_2: g'(i, j) = \left((x_0 x_2 x_4) \cup (x_0 x_4 x_6) \cup (x_0 x_2 x_6) \cup (x_2 x_4 x_6)\right) \bar{g}(i, j)$$

Прилагат се S_1 и S_2, когато анализираният атом има стойност 0. Така, ако логическите условия на дясната страна са истина, се полага $a_{ij} = WHITE$. Иначе $a_{ij} = BLACK$ (кодират се *BLACK* и *WHITE* с произволни стойности, най-често се кодират с 0 и 1).

$$1: g'(i, j) = \left(\bar{x}_0 \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5 \bar{x}_6 \bar{x}_7\right) \bar{g}(i, j)$$

$$2: g'(i, j) = \left(x_1 \cup x_3 \cup x_5 \cup x_7\right) x_0 x_2 x_4 x_6 \; g(i, j)$$

Горните правила се използуват за филтриране на двоични изображения. Двоичният филтър, който работи по част от правилата, се използува в *ipsFilter* за бързо премахване на изолирани петна, диагонални смущения и крайни точки. Функцията *ipsBinaryFilterChunks* реализира пълен двоичен филтър, съгласно всичките дефинирани правила [3].

2.4. Центриране на символ

Центрирането се налага от факта, че несложна невронна мрежа разпознава символната информация само ако символите са центрирани. Алгоритъмът е евристичен и не гарантира центрирането на символа при шумове с размери от порядъка и по-големи от самия символ.

Изходните данни са: двоично изображение с размери $M \times N$. Кодирането е както следва: 1 - бяло; 0 - черно.

Основната идея е да се построят два вектора с размери съответно M и N, които съдържат:

$M(i) = 1$, ако в реда i на $g(i,j)$ има поне една бяла точка, нула в противен случай;

$N(j) = 1$, ако в стълба j на $g(i,j)$ има поне една бяла точка, нула в противен случай.

Предполага се, че символът се центрира във входно поле 7×5. Това означава, че входната матрица е с размери 7×5 и трябва да се построят два вектора, единият от 7 елемента, а другият от 5 елемента, по следния начин:

```
0 0 0 0 0 0
1 0 0 1 0 0
0 0 0 0 0 0
1 0 1 1 0 0
1 0 0 1 0 0
1 0 0 1 0 0
0 0 0 0 0 0
  0 1 1 0 0
```

Така векторът M = (0 1 0 1 1 1 0), а векторът N = (0 1 1 0 0). Сега се трасират векторите M и N, като в тях се търсят най-дългите поредици от единици. По този начин се центрира във входното поле това, което се счита, че е символ (обикновено винаги е с по-големи геометрични размери от шумовете). Ако обаче поради дефекти във възприемащата апаратура се е получил шум с геометрични размери по-големи от размерите на символа, тогава много вероятно е да не се центрира правилно символът и евентуално след това да не бъде разпознат. Функцията, реализираща тази идея, е *ispCenter*.

3. ПРОГРАМНА РЕАЛИЗАЦИЯ И АНАЛИЗ НА РЕЗУЛТАТИТЕ

Всичките програми са написани на език от високо ниво - С. Езикът С е избран за базов, понеже кодът му е лесно преносим и ефективен.

Подробният анализ на резултатите показва, че невронната мрежа постига висок процент на разпознаваемост, ако подадените ѝ изображения са добре филтрирани и центрирани. Входните изображения са силно зашумени. Затова се налага филтриране. Ефектът от работата на филтрите е осезаем. След извършване на локално двоично филтриране на отделните цифри коефициентът на разпознаването нараснa с 5%. Филтърът "изостряне" на изображението повишава коефициента на разпознаване с около 20%.

BEBUG: box before:

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0
0 0 0 0 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 0 0 0
```

BEBUG: box after:

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

ЛИТЕРАТУРА

1. Гочев, Г. Компютърно зрение, ТУ, София, 1993.

2. Барч, Х. Математически формули, Наука и изкуство, преводна, София, 1986.

3. Маргаритов, А. Изследване и разработка на системи от структурно-лингвистичен тип за обработка и разпознаване на визуални изображения, Кандидатска дисертация, ВМЕИ, София, 1989.

АВТОРИ

Гочо Вълев Гочев, доц. д-р, катедра ПИИС, ТУ – София

Цветан Митков Шалявски, инж. аспирант, Национална лаборатория по компютърна вирусология, Българска академия на науките

Иван Трифонов Търнавски, експерт, Национална Лаборатория по Компютърна Вирусология, Българска Академия на Науките