Распознавание одномерных сигналов с использованием OpenCV

(C) 2013. Кручинин А.Ю. http://recog.ru

Описаны некоторые принципы обработки одномерных сигналов средствами OpenCV.

1. Понятие матриц в OpenCV

Для описания изображений, одномерных сигналов в OpenCV предусмотрено понятие матриц, которые определяются структурой CvMat (класс Mat для C++). Для создания матрицы предназначена следующая функция.

```
CvMat* cvCreateMat(
        int rows,
        int cols,
        int type
);
```

Параметры:

rows

Число строк.

cols

Число столбцов.

type

Тип элемента матрицы, который формируется следующим образом: CV_<bit depth><S|U|F> C <number of channels>, где S=signed, U=unsigned, F=float. Например, CV_8UC1 означает: 8битный, беззнаковый с одним каналом, а CV_32SC2 – 32битный, знаковый с 2-мя каналами.

Функция возвращает указатель на матрицу, пример вызова:

```
CvMat* Mat = cvCreateMat( 1, 100, CV_8UC1 );
```

Описанная выше строчка создает одномерную матрицу из 100 элементов. Это не единственный способ создания матриц, но другие способы не будут описываться.

Для обращения к элементам матрицы достаточно обратиться к элементу data структуры CvMat :

```
Mat->data.ptr[5] = 1;
```

Это обращение по указателю unsigned char. Другие указатели можно посмотреть в самой структуре:

```
typedef struct CvMat{
        int type; // Тип
        int step; // Шаг матрицы
        int* refcount;
        union {
                uchar* ptr;
                short* s;
                int* i;
                float* fl;
                double* db;
        } data; // Указатели к данным
        union {
                int rows;
                int height;
        };
        union {
                int cols;
                int width;
        };
} CvMat;
```

Нельзя забывать освобождать память от матрицы:

```
cvReleaseMat( &Mat );
```

Для обозначения матриц используются и другие структуры данных. Так структура IplImage, описывающая изображение является производной от CvMat, а CvArr – некоторая абстрактная структура, которая часто встречается в определениях функций, вместо ней необходимо указывать идентификаторы определенные как CvMat или IplImage.

2. Обработка одномерных сигналов

Что такое одномерный сигнал – последовательность значений определенной скалярной величины. Эти величины могут быть измерены во времени или пространстве. Теория цифровой обработки сигналов хорошо проработана и ее можно найти в ряде работ. В этом параграфе будут рассмотрены некоторые функции OpenCV, которые можно использовать для предварительной обработки сигналов.

cvSmooth

Функция предназначена для сглаживания сигналов и изображений.

```
void cvSmooth(
    const CvArr* src,
    CvArr* dst,
    int smoothtype=CV_GAUSSIAN,
    int param1=3,
    int param2=0,
    double param3=0,
    double param4=0
);
```

Параметры:

src

Исходное изображение (матрица).

dst

Результирующее изображение (матрица).

smoothtype

Тип размытия:

CV_BLUR_NO_SCALE определяет простое размытие без масштабирования, осуществляет суммирование области param1 x param2 без масштабирования;

CV_BLUR определяет простое размытие, осуществляет суммирование области param1 x param2 с последующим масштабированием на величину 1/(param1 * param2);

CV_GAUSSIAN определяет размытие по Гауссу, осуществляет суммирование области param1 x param2 без масштабирования;

CV_MEDIAN определяет медианное размытие, осуществляет поиск среднего значения в области param1 x param1.

CV_BILATERAL определяет двустороннее размытие в области param1 x param1 с цветовой сигмой = param3 и пространственной сигмой = param4.

param1

Ширина области. Значение должно быть нечетное 1,3,5…

param2

Высота области. Значение должно быть нечетное 1,3,5…

param3

Можно указать при CV_GAUSSIAN как значение среднего квадратичного отклонения, если этот параметр равен 0, то среднее квадратичное отклонение вычисляется по следующей формуле:

$$\sigma = 0.3\left(\frac{n}{2} - 1\right) + 0.8, \quad (1)$$

где **n** = **param1** для горизонтали, и **param2** для вертикали.

В первую очередь функция предназначена для обработки изображений, но ей также можно обработать и одномерные сигналы. Пусть есть одномерный сигнал (Рис. 1).

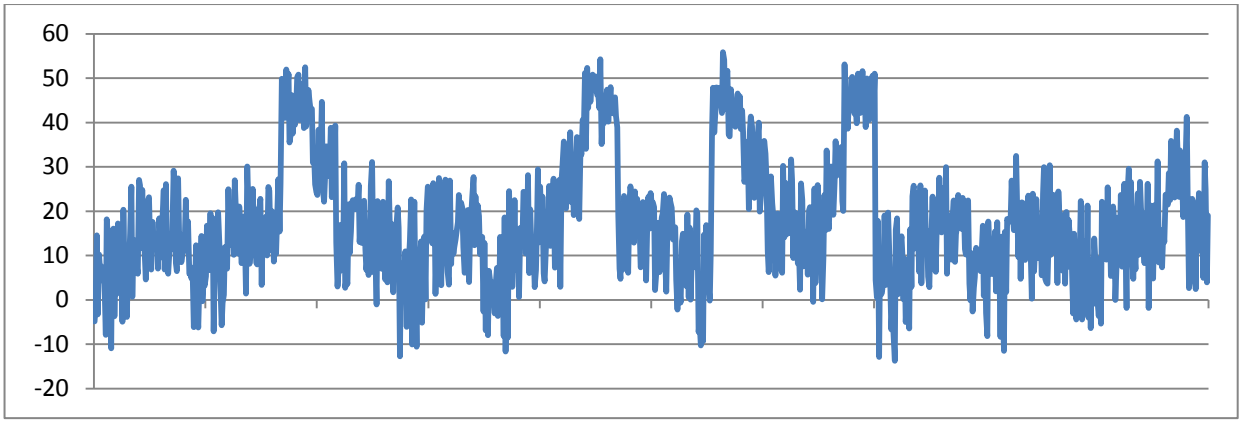

Рис. 1. Одномерный сигнал

Для сглаживания сигнала можно воспользоваться программой, листинг которой приведен ниже. На рисунке 2 представлен результат сглаживания.

Листинг 2.1. Сглаживание одномерного сигнала

```
#include "opencv2/core/core_c.h"
#include "opencv2/imgproc/imgproc_c.h"
#include <stdio.h>

int main()
{
	// Чтение данных из файла
	FILE *f = fopen( "signal1.txt", "r" );
	if ( f == NULL ) return 1;
	int i;
	char buf[10];
	CvMat* Mat = cvCreateMat( 1, 1000, CV_32FC1 );
	for( i = 0; i < 1000; i++ )
	{
		fgets( buf, 10, f );
		Mat->data.fl[i] = atof( buf );
	}
	fclose(f);
	// Сглаживание
	cvSmooth( Mat, Mat, CV_GAUSSIAN, 5 );
	// Запись в результирующий файл
```

```
        f = fopen( "signal1_out.txt", "w" );
        if ( f == NULL ) return 2;
        for( i = 0; i < 1000; i++ )
        {
              sprintf( buf, "%5.3f\n", Mat->data.fl[i] );
              fputs( buf, f );
        }
        fclose(f);
        cvReleaseMat( &Mat );
        return 0;
}
```

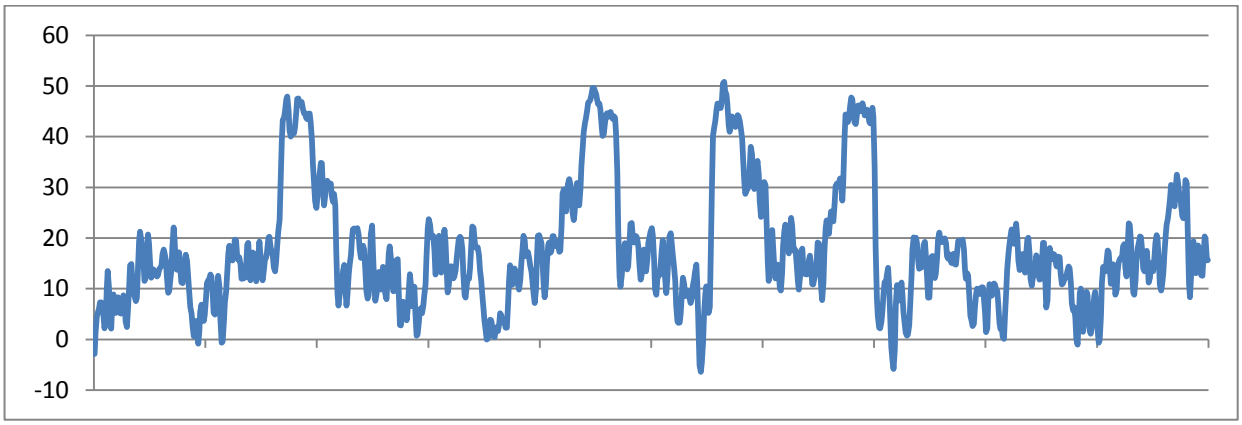

Рис. 2.2. Сглаженный сигнал

При обработке сигналов можно использовать и морфологические операции, которые используются для обработки изображений.

cvErode

Функция эрозии.

```
void cvErode(
        const CvArr* src,
        CvArr* dst,
        IplConvKernel* element=NULL,
        int iterations=1
);
```

Параметры:

src

Исходное изображение (матрица).

dst

Результирующее изображение (матрица).

element

Структурный элемент, используемый для эрозии. Если равен NULL, то используется 3 x 3 элемент по умолчанию.

iterations

Количество применяемых операций эрозии.

Для того, чтобы посмотреть результат работы эрозии, в пример 2.1 внесем следующие изменения – вместо операции сглаживания вставим следующие команды:

```
cvErode(Mat, Mat );
```

Результат работы в этом случае показан на рисунке 3.

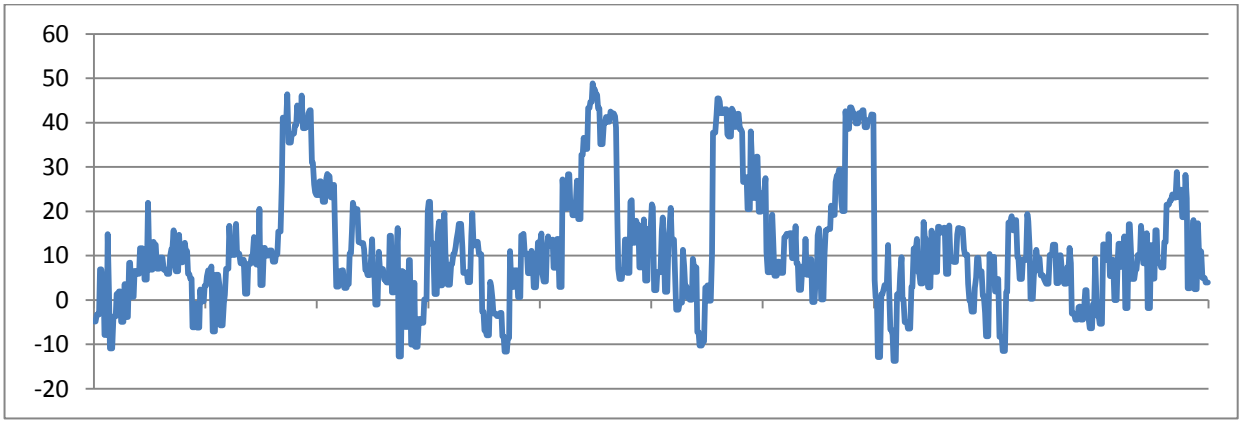

Рис. 3. Сигнал обработанный cvErode

Функция насыщения cvDilate имеет такой же формат. Результат работы показан на рисунке 4.

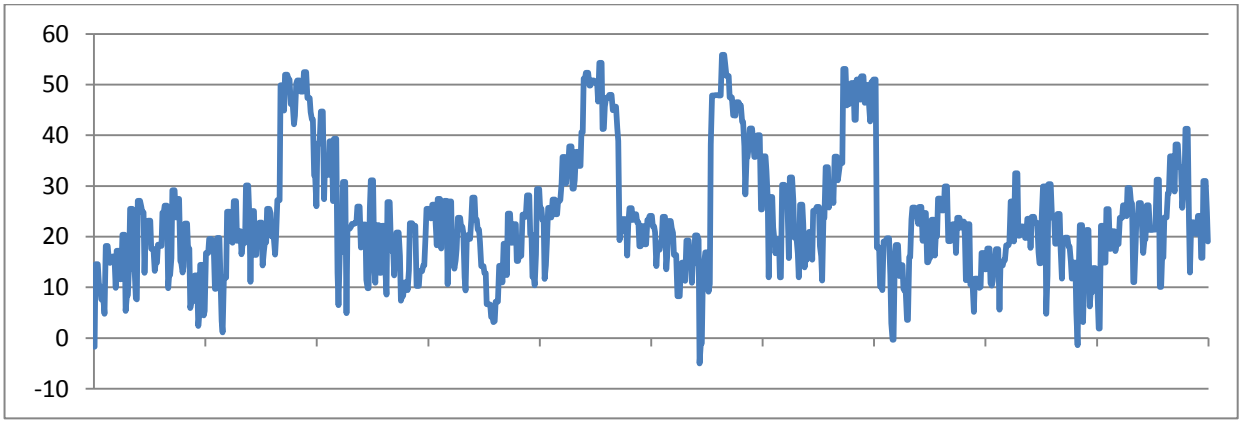

Рис. 4. Сигнал обработанный cvDilate

Важным при работе с одномерными сигналами является передискретизация (изменение размеров). Для этого можно также воспользоваться функцией, предназначенной для изменения размеров изображений.

cvResize

Функция передискретизации/интерполяции (растяжения/сжатия) сигналов и изображений.

void cvResize(

const CvArr* src,

CvArr* dst,

int interpolation=CV_INTER_LINEAR

);

Параметры:

src

Исходное изображение (матрица).

dst

Результирующее изображение (матрица).

interpolation

Метод интерполяции: CV_INTER_NN – метод ближайшего соседа; CV_INTER_LINEAR – билинейная (по умолчанию); CV_INTER_AREA – интерполяция на основе соотношений площади пикселей; CV_INTER_CUBIC – бикубическая.

Для проверки работы функции изменим код листинга 1 следующим образом (Листинг 2).

Листинг 2. Передискретизация сигнала

```
#include "opencv2/core/core_c.h"
#include "opencv2/imgproc/imgproc_c.h"
#include <stdio.h>

int main()
{
	// Чтение данных из файла
	FILE *f = fopen( "signal1.txt", "r" );
	if ( f == NULL ) return 1;
	int i;
	char buf[10];
	CvMat* Mat = cvCreateMat( 1, 1000, CV_32FC1 );
	CvMat* Mat2 = cvCreateMat( 1, 100, CV_32FC1 );
	for( i = 0; i < 1000; i++ )
	{
```

```
            fgets( buf, 10, f );
            Mat->data.fl[i] = atof( buf );
      }
      fclose(f);
      // Изменение размеров
      cvResize(Mat, Mat2 );
      // Запись в результирующий файл
      f = fopen( "signal1_out.txt", "w" );
      if ( f == NULL ) return 2;
      for( i = 0; i < 100; i++ )
      {
            sprintf( buf, "%5.3f\n", Mat2->data.fl[i] );
            fputs( buf, f );
      }
      fclose(f);
      cvReleaseMat( &Mat );
      cvReleaseMat( &Mat2 );
      return 0;
}
```

Результат передискретизации показан на рисунке 5.

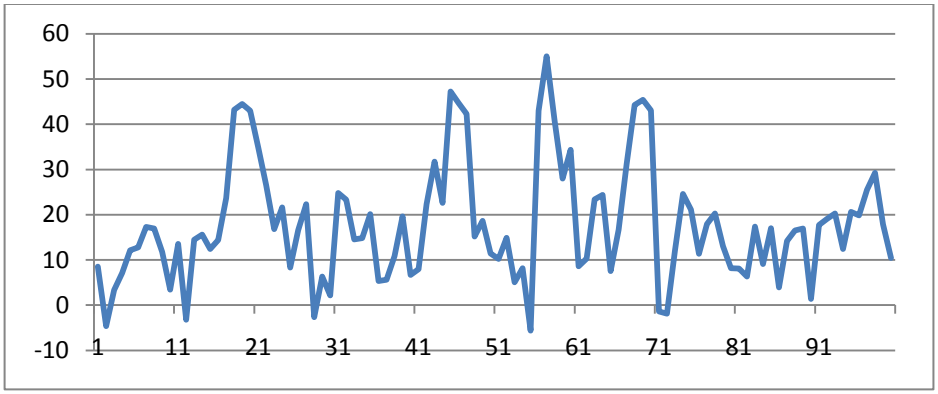

Рис. 5. Сигнал, уменьшенный в размерах в 10 раз

Также важной функцией для обработки сигналов является морфологический градиент, позволяющий выявить перепады сигналов. Для иллюстрации заменим сглаживание в листинге 1. на следующие строки:

```
int r=3;
int c=1;
IplConvKernel* structuringElement = cvCreateStructuringElementEx(r, c,
      cvFloor(r / 2), cvFloor(c/ 2), CV_SHAPE_RECT, NULL);
CvMat* Mattmp = cvCreateMat( 1, 1000, CV_32FC1 );
cvMorphologyEx( Mat, Mat, Mattmp,structuringElement,CV_MOP_GRADIENT,1);
```

```
cvReleaseMat( &Mattmp );
```

На рисунке 6 приведен результат вызова морфологического градиента.

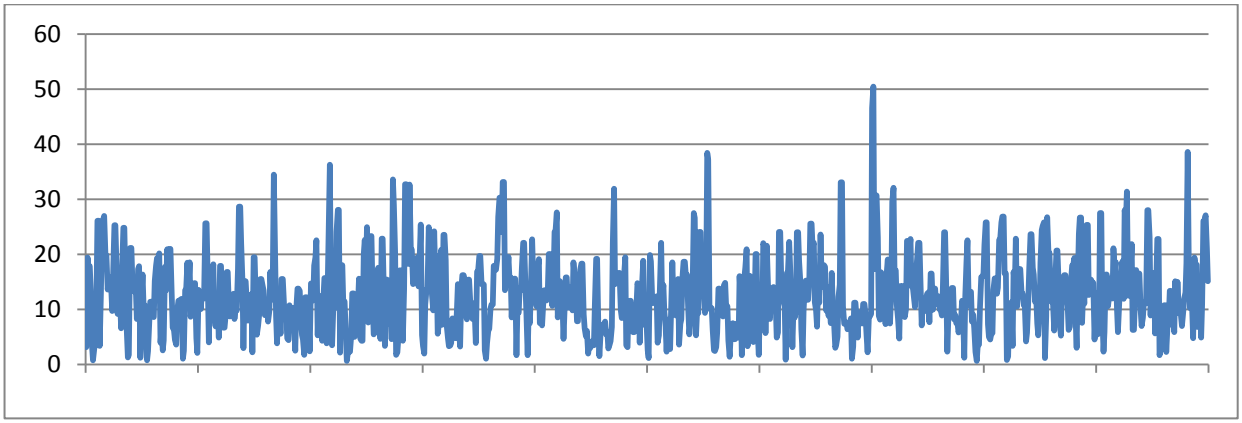

Рис. 6. Морфологический градиент

Помимо описанных функций существуют и другие технологии обработки сигнала.

3. Преобразование Фурье

Для получения спектра сигнала предназначено преобразование Фурье. Прежде всего преобразование предназначено для обработке непрерывных сигналов, но его математический аппарат применим и для дискретных сигналов. При этом в вычислениях используется быстрое преобразование Фурье (БПФ). Прямое преобразование Фурье:

$$\dot{X}(n) = \sum_{k=0}^{N-1} x(k)\exp\left(-j\frac{2\pi nk}{N}\right). \quad (2)$$

Существует обратное преобразование Фурье (из дискретного спектра к сигналу):

$$x(k) = \frac{1}{N}\sum_{n=0}^{N-1} \dot{X}(n)\exp\left(j\frac{2\pi nk}{N}\right). \quad (3)$$

Для преобразования Фурье в OpenCV предусмотрена следующая функция.

```
void cvDFT(
    const CvArr* src,
    CvArr* dst,
    int flags,
    int nonzeroRows=0
);
```

src

Исходное изображение (матрица).

dst

Результирующее изображение (матрица).

flags

Флаги трансформации.

CV_DXT_FORWARD – прямое преобразование Фурье.

CV_DXT_INVERSE – обратное преобразование.

CV_DXT_SCALE – масштабируемый результат: делится на число элементов массива.

CV_DXT_ROWS – преобразование используется для каждого вектора отдельно. Применяется для обработки многократных векторов одновременно.

CV_DXT_INVERSE_SCALE – CV_DXT_INVERSE вместе с CV_DXT_SCALE.

nonzeroRows

Количество рядов отличных от нуля в исходном множестве. Предназначено для ускорения преобразований. Если значение меньше 1, то игнорируется.

Для получение спектра сигнала над матрицей достаточно осуществить следующую операцию:

```
cvDFT( Mat, Mat, CV_DXT_FORWARD );
```

Результат работы показан на рисунке 7.

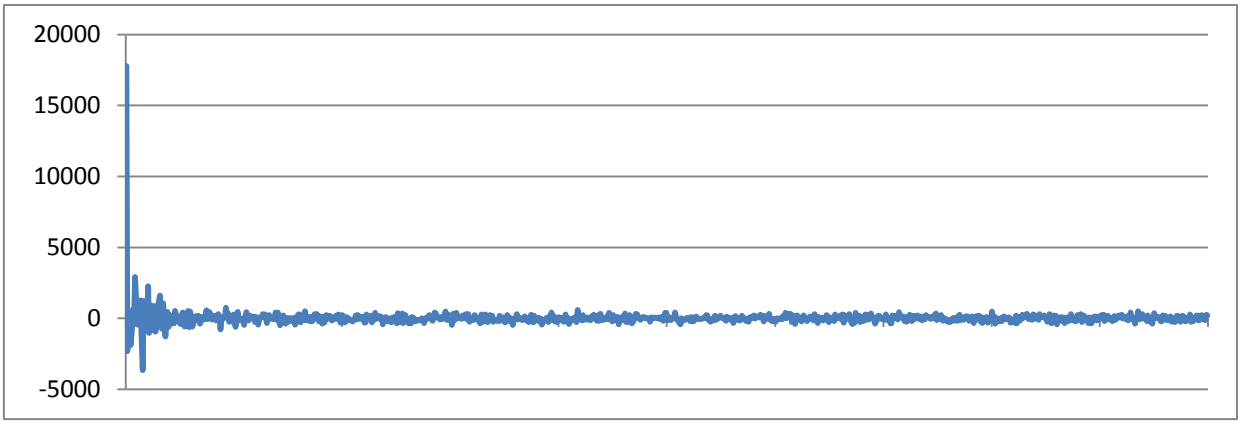

Рис. 7. Спектр сигнала, представленного на рисунке 1

Спектральный анализ широко используется для распознавания одномерных сигналов, в частности для распознавания речи.

4. Сравнение сигналов с шаблоном

Для сравнения сигнала с шаблоном обычно используется корреляционный анализ, который определяет степень «похожести» сигналов друг на друга. Критерием, отличающим образы, является коэффициент парной корреляции:

$$K = \frac{K_{xy}}{\sigma_x \sigma_y}, K_{xy} = \frac{1}{N}\sum_{n=1}^{N}[x(n) - M_x][y(n) - M_y], \quad (4)$$

где M_x и M_y – оценки математического ожидания для переменных *x* и *y*; K_{xy} – коэффициент ковариации; σ_x и σ_y – оценки среднего квадратичного отклонения; *N* – число элементов исследуемых рядов.

В OpenCV есть функция, предназначенная для сравнения с шаблоном матриц/изображений не только корреляционным способом, но и другими.

```
void cvMatchTemplate(
    const CvArr* image,
    const CvArr* templ,
    CvArr* result,
    int method
);
```

Параметры:

image

Матрица/изображение, где будет осуществляться поиск (8 битное или 32-битное floating-point);

templ

Искомый шаблон матрицы/изображения, не больше чем **image** и того же типа;

result

Карта результатов сравнения, одноканальный 32-битный массив с плавающей точкой. Если размеры **image** WxH, а **templ** wxh, то размер **result** (W-w+1)x(H-h+1);

method

Метод сравнения шаблона.

Функция скользит по матрице/изображению, сравнивая перекрывающиеся участки размером wxh с **templ** используя метод **method**, сохраняя результат в **result**. Ниже описаны формулы для сравнения (*I* – **image**, *T* – **templ**, *R* – **result**). Суммирование проводится по шаблону и/или по региону матрицы/изображения по следующему пути: x' = 0..w-1, y' = 0..h-1.

method = CV_TM_SQDIFF – квадрат отклонения

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2. \quad (5)$$

method = CV_TM_SQDIFF_NORMED – квадрат отклонения с нормированием

$$R(x,y) = \frac{\sum_{x',y'}(T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 * \sum_{x',y'} I(x+x',y+y')^2}}. \quad (6)$$

method = CV_TM_CCORR – корреляция результатов

$$R(x,y) = \sum_{x',y'} T(x',y') * I(x+x',y+y'). \quad (7)$$

method = CV_TM_CCORR_NORMED – корреляция результатов с нормированием

$$R(x,y) = \frac{\sum_{x',y'} T(x',y') * I(x+x',y+y')}{\sqrt{\sum_{x',y'} T(x',y')^2 * \sum_{x',y'} I(x+x',y+y')^2}}. \quad (8)$$

method = CV_TM_CCOEFF – другой способ корреляции

$$R(x,y) = \sum_{x',y'} T'(x',y') * I'(x+x',y+y'), \quad (9)$$

где

$$T'(x',y') = T(x',y') - \frac{1}{w*h*\sum_{x'',y''} T(x'',y'')}, \quad (10)$$

$$I'(x+x',y+y') = I(x+x',y+y') - \frac{1}{w*h*\sum_{x'',y''} I(x+x'',y+y'')}$$

method = CV_TM_CCOEFF_NORMED – другой способ корреляции с нормализацией

$$R(x,y) = \frac{\sum_{x',y'} T'(x',y') * I'(x+x',y+y')}{\sqrt{\sum_{x',y'} T'(x',y')^2 * \sum_{x',y'} I'(x+x',y+y')^2}}. \quad (11)$$

Формула нормализации (знаменатель выражений (2.6), (2.8), (2.11)) разработаны Гальтоном и описаны Роджерсом.

Для того, чтобы проверить работу функции, используем тот же сигнал, что и в предыдущих примерах, только сглаженный (Рис. 2). А в качестве шаблона вырежем из этого сигнала 60 значений, начиная с 170 элемента. Листинг примера

Листинг 3. Поиск шаблона в сигнале

```
#include "opencv2/core/core_c.h"
#include "opencv2/imgproc/imgproc_c.h"
#include <stdio.h>

int main()
{
	// Чтение данных из файла
	FILE *f = fopen( "signal1.txt", "r" );
	if ( f == NULL ) return 1;
	int i;
	char buf[100];
	CvMat* Mat = cvCreateMat( 1, 1000, CV_32FC1 );
	for( i = 0; i < 1000; i++ )
	{
```

```
            fgets( buf, 10, f );
            Mat->data.fl[i] = atof( buf );
        }
        fclose(f);
        // Сглаживание
        cvSmooth( Mat, Mat, CV_GAUSSIAN, 5 );
        // Шаблон
        CvMat* Mattempl = cvCreateMat( 1, 60, CV_32FC1 );
        for( i = 0; i < 60; i++ ) Mattempl->data.fl[i] = Mat->data.fl[i+170];
        CvMat* MatResult = cvCreateMat( 1, 1000-60+1, CV_32FC1 );
        cvMatchTemplate( Mat, Mattempl, MatResult, CV_TM_SQDIFF_NORMED );
        // Запись в результирующий файл
        f = fopen( "signal1_out.txt", "w" );
        if ( f == NULL ) return 2;
        for( i = 0; i < 1000-60+1; i++ )
        {
            sprintf( buf, "%5.3f\n", MatResult->data.fl[i] );
            fputs( buf, f );
        }
        fclose(f);
        cvReleaseMat( &Mat );
        cvReleaseMat( &Mattempl );
        cvReleaseMat( &MatResult );
        return 0;
}
```

На рисунках 8 – 10 представлены результаты сравнения методами CV_TM_SQDIFF_NORMED, CV_TM_CCORR_NORMED и CV_TM_CCOEFF_NORMED соответственно.

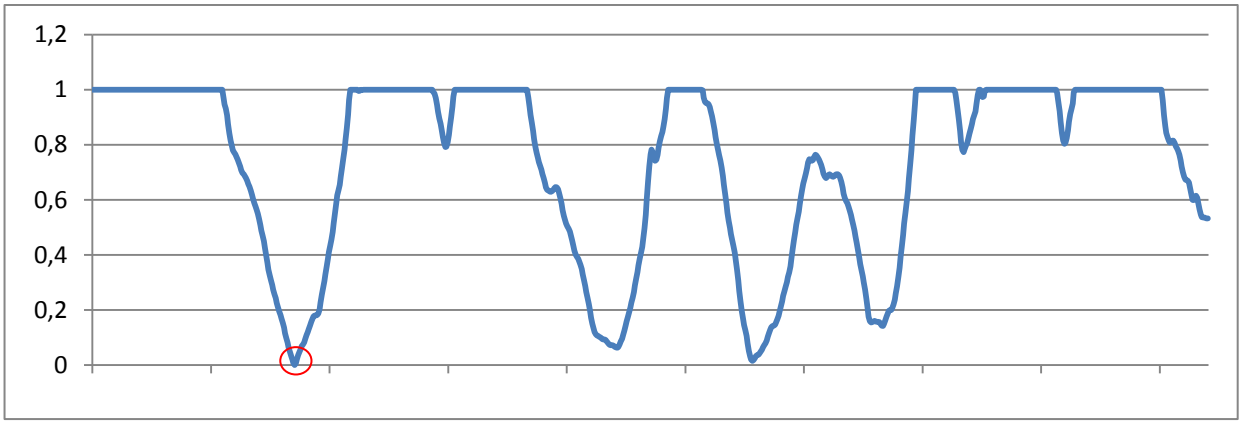

Рис. 8. Результирующая матрица методом CV_TM_SQDIFF_NORMED

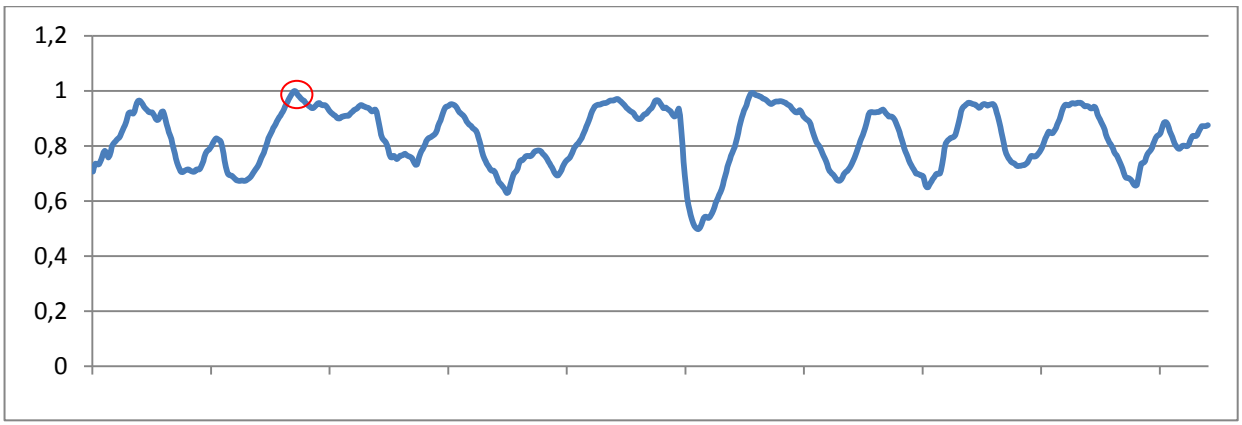

Рис. 9. Результирующая матрица методом CV_TM_CCORR_NORMED

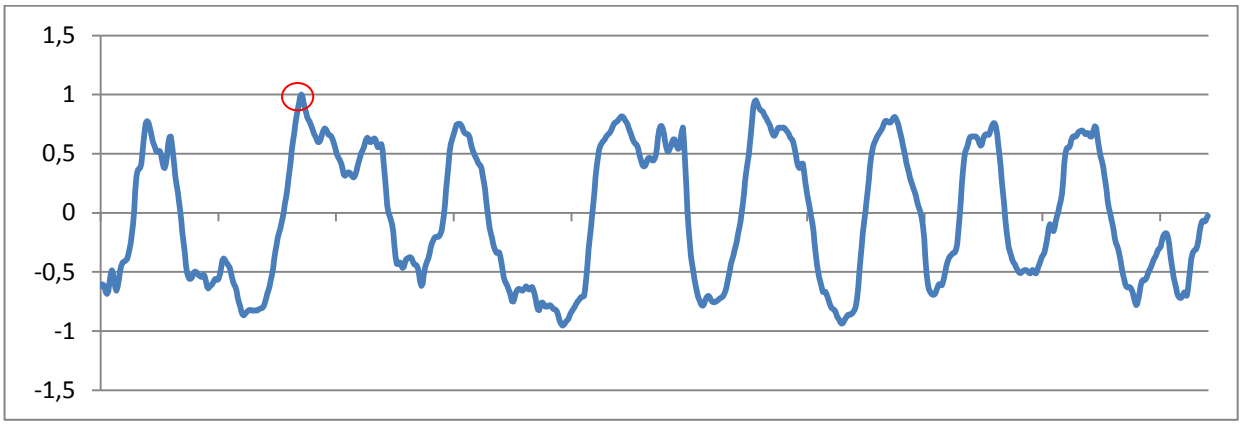

Рис. 10. Результирующая матрица методом CV_TM_CCOEFF_NORMED

Для метода CV_TM_SQDIFF_NORMED наилучшим совпадением является близость к 0, а в других двух методах к 1. На графиках показаны точки максимального совпадения с эталоном.

Заключение

В данном материале описаны только некоторые возможности обработки и распознавания одномерных сигналов, что может помочь начинающим разработчикам/исследователям быстрее овладеть возможностями OpenCV по обработке и распознаванию сигналов.