# Контура

Шокуров Антон В.
shokurov.anton.v@yandex.ru
http://машинноезрение.рф

22 октября 2018 г.

Версия: 0.09

**Аннотация**

Преобразование растровых изображений в бинарные. Пороговые функции. Связанные компоненты. Их подсчет.

## 1 Контура

В данной заметке будет показано как выделять контура у объектов на бинарном изображении. Далее показано, как их аппроксимировать аналитическими кривыми: линия, окружность и тому подобное.

### 1.1 Выделение контуров

Выделение контуров и их иерархию. В целом для их выделения используется функция findContours, но в зависимости от конкретной задачи с разными параметрами.

Напомню, что мы придерживаемся идеологии, что у нас есть фон и есть объекты. У объекта можно выделить внешний контур, а также внутренний (там где дырки). В общем случае можно построить иерархию объектов.

**Выделение внешних контуров** Напомню, как считать и превратить цветное изображение в бинарное:

```
1 Mat color;
2 color = imread(argv[1], IMREAD_COLOR);
3
```

```
4 Mat gray;
5 cvtColor( color, gray, COLOR_BGR2GRAY );
6
7 Mat bin;
8 threshold(gray, bin, 180, 255, THRESH_BINARY_INV );
```

Далее, для выделения только внешних контуров на изображение используется такой код: что это делается так:

```
1 vector<vector<Point2i>> contours;
2 vector<Vec4i> hierarchy;
3 findContours( bin, contours, hierarchy,
4     RETR_EXTERNAL, CHAIN_APPROX_SIMPLE );
```

Наиболее важна для нас переменная `contours`. Переменная `hierarchy` будет использована в следующих параграфах.

В `contours` хранятся выделенные контура объекта. При вызове было задано `RETR_EXTERNAL` поэтому будут выделены только внешние контура объектов.

Параметр `CHAIN_APPROX_SIMPLE` указывает на то, что подряд идущие точки по горизонтали и вертикали не следует записывать в контур (для экономии памяти). Если указать значение `HAIN_APPROX_NONE` , то будут указаны все точки контура (согласно идеологии 8-связанности).

Итого, размер вектора `contours` указывает на количество связанных компонент.

```
1 cout << "компоненты: " << contours.size() << endl;
```

Исходя их определения переменной `contours` следует, что её элементы в свою очередь также являются векторами типа `vector<Point2i>`, т.е. состоят из точек. Поэтому, `contours[i][j]` задает координаты $j$ой точки $i$го контура.

Упражнение. Напиши программу, которая отрисовывает контура (разными цветами).

**Отрисовка контуров** Отрисовать контура можно и использую, соответствующую, функцию опенсв: `drawContours`.

```
1 drawContours( color, contours, -1, Scalar( 0, 100, 200), 2 );
```

Первый аргумент указывает на изображение на котором будет выполнена отрисовка контуров. Второй аргумент задает сами контура. Третий на индекс контура, который нужно отрисовать. В случае, если задать число $-1$ будут отрисованы все контура. Далее задается цвет и толщина линий (отрезков).

**Выделение внутренних контуров** Далее рассмотрим как выделять дырки в объектах. Заменем параметр `RETR_EXTERNAL` на `RETR_CCOMP`.

```
findContours( bin, contours, hierarchy,
    RETR_CCOMP, CHAIN_APPROX_SIMPLE );
```

Последний параметр позволяет создать простейшую иерархию их контуров. Он помимо внешнего контура объекта выделяет и внутренние, т.е. ищет дырки. Данная информация занесена в переменную `hierarchy`. Рассмотрем её подробнее для понимания того как это происходит.

Каждый из элементов вектора `hierarchy` (имея тип `Vec4i`) состоит из 4 целочисленных компонент. Он имеет такой же размер, что и вектор контуров (`contours`). Фактически в переменной `hierarchy` для каждого контура хранится его положение в иерархии контуров, т.е. какие у него соседние дыры (братья/сестры), дыры (дети) и родитель. Так, все компоненты являются индексами. Индекс равный $-1$ означает отсутствие данной сущности. Итого, компоненты указывают на индекс:

- 0я – следующего брата/сестру;
- 1я – предыдущего брата/сестру;
- 2я – ребенка;
- 3я – родителя.

В случае использования параметра `RETR_CCOMP` иерархия устроена просто. Для каждого внешнего контура указаны контура дыр в случае их наличия.

```
int cntrs = contours.size();
cout << "контуры: " << cntrs << endl;
int i, k = 0;
for( i = 0; i < cntrs && i != -1 ; i = hierarchy[i][0], k++ )
{
  const Vec4i &data = hierarchy[i];
  int child = data[2], holes = 0;
  int j;
  cout << "контур " << k << ": ";
  for( j = child; j != -1; j = hierarchy[j][0], holes++ );
  if( holes )
    cout << "дыры: " << holes;
  cout << endl;
}
```

**Определение иерархии** Для полноценного разбора всех контуров необходимо заменить в первоначальном вызове параметр `RETR_EXTERNAL` на `RETR_TREE`, т.е. сделать вызов следующим образом:

```
findContours( bin, contours, hierarchy,
    RETR_TREE, CHAIN_APPROX_SIMPLE );.
```

При таком вызове в переменной `hierarchy` хранится полноценная информация об иерархии контуров: в ней для каждой дыры хранится контура которые в ней находятся.

Напишем функцию, которая печатает эту иерархию:

```
void print_tree( const vector<Vec4i> &hierarchy, int s)
{
  int cntrs = hierarchy.size();
  int i;
  cout << "[ ";
  for( i = s; i < cntrs && i != -1 ; i = hierarchy[i][0])
  {
    if( i != s )
      cout << ",";
    cout << " ";
    const Vec4i &data = hierarchy[i];
    print_tree( hierarchy, data[2] );
  }
  if( i != s )
    cout << " ";
  cout << " ]";
}
```

Вызвать эту функцию следует следующим образом:

```
print_tree( hierarchy, 0 );
```

Упражнение. Напишите программу, которая на изображении выделает все контура, которые содержать две дырки и одну дырку с объектом.

## 1.2 Характеристики контуров

Можно применить и моменты пройденные ранее.

**Длины** `arcLength`

**Площадь** `contourArea`

## 1.3 Аппроксимация кривых

После того как контура выделены их можно аппроксимировать аналитическими кривыми.

`approxPolyDP`

**Линия** `fitLine`

**Окружности** `fitEllipse`

**Эллипс** `fitEllipseAMS` `fitEllipseDirect`

## 1.4 Вокруг множество точек

`boundingRect` `minAreaRect`
`minEnclosingCircle` `minEnclosingTriangle`