OpenCV : How to find the pixels inside a contour in c++

Asked 3 years, 10 months ago Active 2 years, 5 months ago Viewed 8k times

5

3

Suppose if we are working on an image, is there any way to access the pixels inside the contour?

I have already found the contour using the function findContours() and even found the moments but I couldn't find the pixels inside the contour.

Any suggestions are Welcome!!

Thank you!

c++ opencv opencv-contour

asked Sep 29 '16 at 12:29
Bloklo
61 1 4

What exactly do you mean by "finding the pixels"? Define your problem more clearly, and even better illustrate what you mean by providing some example. – Dan Mašek Sep 29 '16 at 12:48

Actually, I am getting the pixels of the boundary by using the function findContours() but I am not able to get the pixels inside the contour or boundary. I need to find the pixels without iterating through the whole image pixels. – Bloklo Sep 29 '16 at 12:57

2 Can't you use `connectedComponents` in the first place? – Miki Sep 29 '16 at 13:05

contours is array of array so the i-th contour is contour[i] try to get the points from contour[i][0] .. contour[i][N] were N = contour[i].size()-1 – Amitay Nachmani Sep 29 '16 at 13:39

@AmitayNachmani Thankyou.. But the method you specified will give the pixels of the contour. But I need the pixels which are enclosed by the contour, ie, the pixels of the object which i have found the contour. – Bloklo Sep 29 '16 at 13:51

3 Answers

Active Oldest Votes

5

As @Miki already mentioned you can use connectedComponents to perform a labeling. Then you iterate through the bounding box of your object like @Amitay Nachmani suggested. But instead of using pointPolygonTest you can check if the value at your current positions matches your current label Here is a small example:

```
#include "opencv2/imgproc.hpp"
#include "opencv2/highgui.hpp"
#include <vector>

using namespace cv;
using namespace std;

Mat binary, labels, stats, centroids;
int main()
{
    Mat src = imread("C:\\Users\\phili\\Pictures\\t06-4.png",0);
    threshold(src, binary, 0, 255, CV_THRESH_OTSU);
    int nLabels = connectedComponentsWithStats(binary, labels, stats,
centroids);
    vector<vector<Point>> blobs(nLabels-1);
    for (int i = 1; i < nLabels; i++) //0 is background
    {
        //get bounding rect
        int left =  stats.at<int>(i, CC_STAT_LEFT) ;
        int top = stats.at<int>(i, CC_STAT_TOP);
        int width = stats.at<int>(i, CC_STAT_WIDTH);
        int height = stats.at<int>(i, CC_STAT_HEIGHT);

        blobs[i - 1].reserve(width*height);
        int x_end = left + width;
        int y_end = top + height;
        for (int x = left; x < x_end; x++)
        {
            for (int y = top; y < y_end; y++)
            {
                Point p(x, y);
                if (i == labels.at<int>(p))
                {
                    blobs[i-1].push_back(p);
                }
            }

        }
    }
}
```

EDIT:

Since youre using OpenCV 2.4 there are two ways to achieve the same results. First you could use findContours to detect the blobs, then draw them (filled) into a new image with a specific color as label (be aware that your blobs could contain holes) Then iterate through the image inside the bounding rectangle of each contour and get all points with the label of your current contour. If you just iterate through the bounding rectangle inside your binary image, you have problems with objects overlapping the bounding rectangle. Here is the code:

```
int getBlobs(Mat binary, vector<vector<Point>> & blobs)
{
    Mat labels(src.size(), CV_32S);
    vector<vector<Point>> contours;
    vector<Vec4i> hierarchy;
    findContours(binary, contours, hierarchy, CV_RETR_CCOMP,
CV_CHAIN_APPROX_NONE);
    blobs.clear();
    blobs.reserve(contours.size());
    int count = 1; //0 is background
    for (int i = 0; i < contours.size(); i++) // iterate through each contour.
    {
        //if contour[i] is not a hole
        if (hierarchy[i][3] == -1)
        {
            //draw contour without holes
            drawContours(labels, contours, i, Scalar(count),CV_FILLED, 0,
hierarchy, 2, Point());
            Rect rect = boundingRect(contours[i]);
            int left = rect.x;
            int top = rect.y;
            int width = rect.width;
            int height = rect.height;
            int x_end = left + width;
            int y_end = top + height;
            vector<Point> blob;
            blob.reserve(width*height);
```

```
                for (size_t x = left; x < x_end; x++)
                {
                    for (size_t y = top; y < y_end; y++)
                    {
                        Point p(x, y);
                        if (count == labels.at<int>(p))
                        {
                            blob.push_back(p);
                        }
                    }
                }
                blobs.push_back(blob);
                count++;
            }

        }
        count--;
        return count;
    }
```

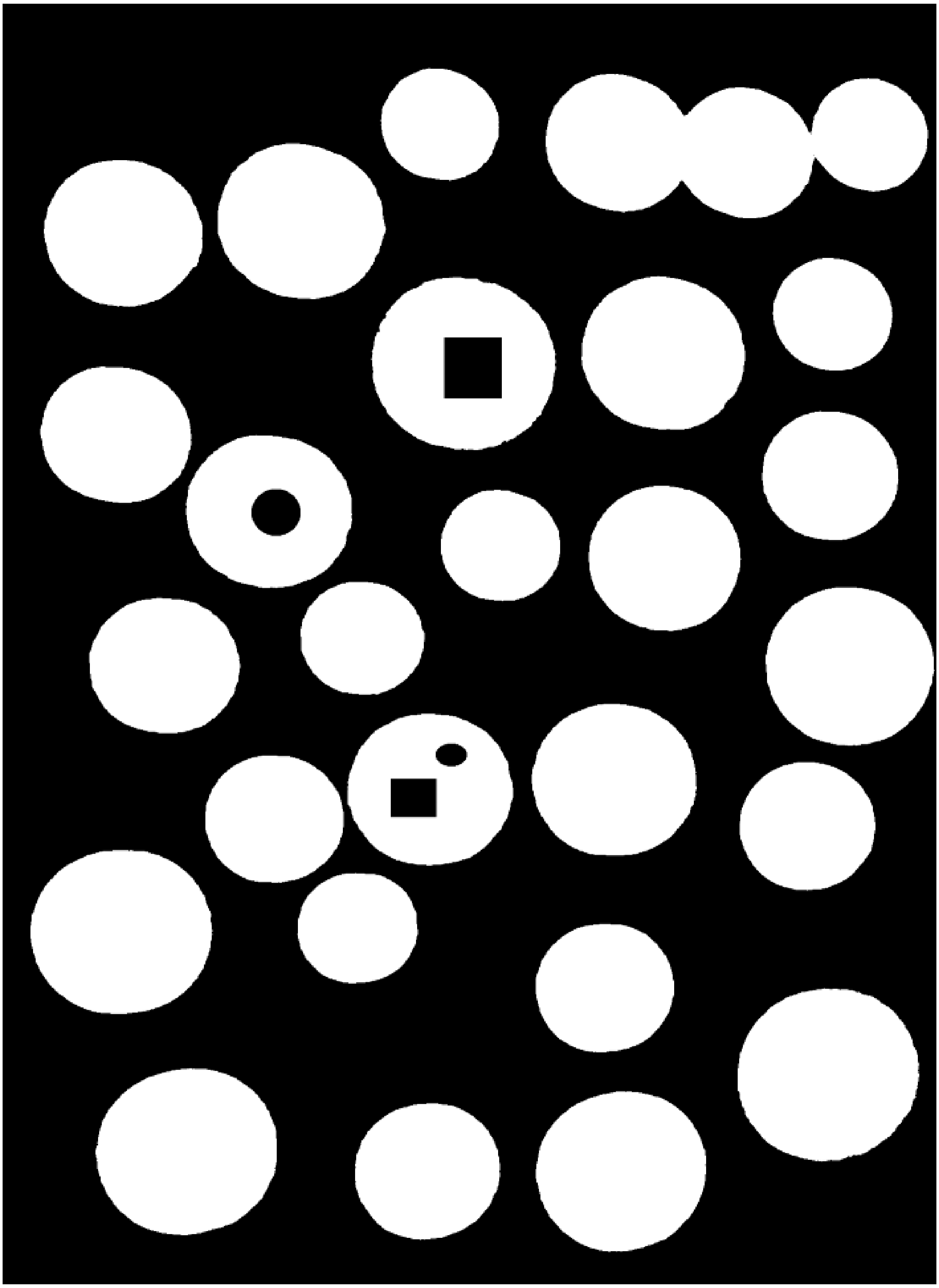
Second you can perform your own labling with floodfill. Therefore you iterate through your image and start floodfill for every white pixel, iterate through the bounding rectangle and get all points that have the same seedColor. Here is the code:

```
int labeling(Mat binary, vector<vector<Point>> &blobs)
{
    FindBlobs(binary, blobs);
    return blobs.size();
}
```

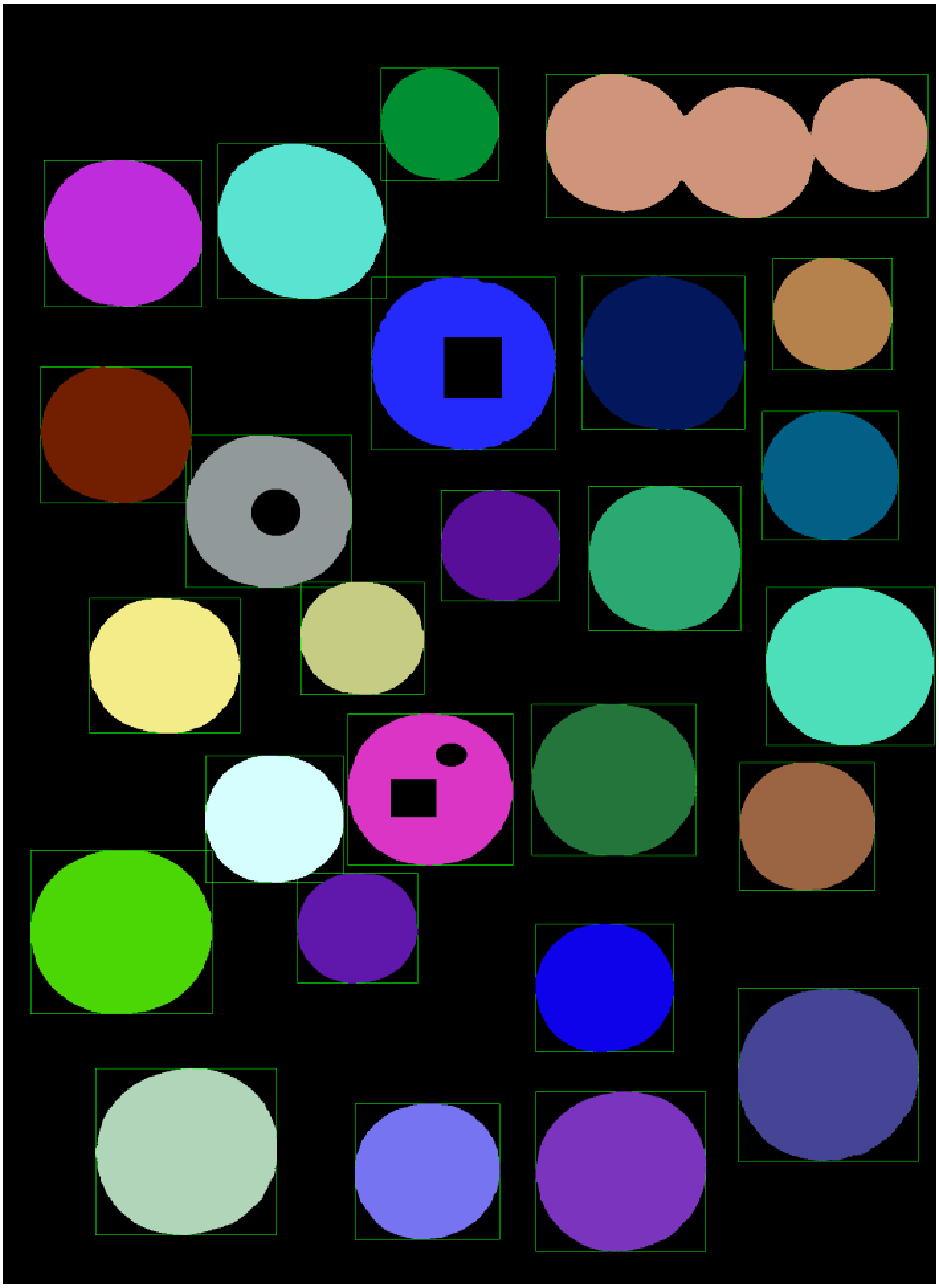
with

```
void FindBlobs(const Mat &binary, vector<vector<Point>> &blobs)
{
    blobs.clear();
    // Fill the label_image with the blobs
    // 0  - background
    // 1  - unlabelled foreground
    // 2+ - labelled foreground
    cv::Mat label_image;
    binary.convertTo(label_image, CV_32FC1);
    float label_count = 2; // starts at 2 because 0,1 are used already
    for (int y = 0; y < label_image.rows; y++) {
        float *row = (float*)label_image.ptr(y);
        for (int x = 0; x < label_image.cols; x++) {
            if (row[x] != 255) {
                continue;
            }
            cv::Rect rect;
            cv::floodFill(label_image, Point(x, y), Scalar(label_count), &rect,
Scalar(0), Scalar(0), 4 );
            vector<Point> blob;
            blob.reserve(rect.width*rect.height);

            for (int i = rect.y; i < (rect.y + rect.height); i++) {
                float *row2 = (float*)label_image.ptr(i);
                for (int j = rect.x; j < (rect.x + rect.width); j++) {
                    if (row2[j] != label_count)
                    {
                        continue;
                    }
                    blob.push_back(Point(j, i));
                }
            }

            blobs.push_back(blob);
            label_count++;
        }
    }
}
```

I used this image:



And here are the bounding boxes and the points inside the contour for visualization:



edited Oct 7 '16 at 10:38

answered Sep 30 '16 at 10:59

PSchn

638 4 12

Thank you!! @PSchn , I am getting some issue saying that "connectedcomponents does not exist". I read somewhere that it is available only in opencv version 3.1 but mine is 2.4 version... And also I read like opencv 3.1 is unstable. – Bloklo Oct 3 '16 at 6:24

In that case you need to use findContours and calculate the bounding rectangle. I will update my answer. But I cant tell you if opencv 3.1 is unstable, sorry – PSchn Oct 3 '16 at 7:12

Sure!! and Thank you! – Blokl o Oct 3 '16 at 7:14

Can you please update your answer by using findContours and calculate the bounding rectangle? – Bloklo Oct 5 '16 at 8:46

Yes i will do it today, sorry i had no time yet – PSchn Oct 5 '16 at 16:57

2

Create a new image with filled contours using fillPoly.

```
    fillPoly(filledImage, contours, Scalar(255, 255, 255));
```

Then find the non-zero pixels within that image using findNonZero.

```
   vector<Point> indices;
   findNonZero(filledImage, indices);
```

The "indices" result refer to pixels inside the contour

answered Mar 23 '18 at 20:04
Lawes
97 3

0

Use the pointPolygonTest http://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=pointpolygontest#pointpolygontest on the all the pixels inside the bounding box of the contour contour.

answered Sep 29 '16 at 14:11
Amitay Nachmani
2,954 1 12 18

Thank you!! but this will give, whether a pixel is inside the contour or outside or on the contour. But what is required is, a list of all the pixels inside the contour so that I can randomly select a pixel from it for my further calculations. – Bloklo Sep 29 '16 at 15:54

So like i said go over all the pixels that are in the bounding box of the counter and check for each one of them if it is inside or not. If yes add it to a vector of pixels that are inside – Amitay Nachmani Sep 29 '16 at 16:36

Yes.. Amitay.. even I thought so.. but I was thinking something different like, without iterating throughout the whole image pixels, is there any other method to access the pixels inside? – Bloklo Sep 29 '16 at 16:41

why not choose random pixel positions inside the bounding box and check if position is inside contour? Even faster if you do it in a labeled image (e.g. from connected components) and check if value position is not equal your background value or equal your actual label. – PSchn Sep 29 '16 at 18:55