OpenCV: Find all non-zero coordinates of a binary Mat image

Asked 6 years, 10 months ago Active 2 years, 11 months ago Viewed 33k times

20

7

I'm atttempting to find the non-zero (x,y) coordinates of a binary image.

I've found a few references to the function `countNonZero()` which only counts the non-zero coordinates and `findNonZero()` which I'm unsure how to access or use since it seems to have been removed from the documentation completely.

This is the closest reference I found, but still not helpful at all. I would appreciate any specific help.

Edit: - To specify, this is using C++

c++ opencv image-processing computer-vision computer-science

edited May 23 '17 at 12:16
Community ♦
1 1

asked Oct 8 '13 at 8:25
DMor
727 2 8 16

1 `findNonZero()` description is within operations on array section currently. – Leonid Vasilev Jul 21 '17 at 8:58

3 Answers

Active Oldest Votes

35

Here is an explanation for how `findNonZero()` saves non-zero elements. The following codes should be useful to access non-zero coordinates of your **binary** image. Method 1 used `findNonZero()` in OpenCV, and Method 2 checked every pixels to find the non-zero (positive) ones.

Method 1:

```
#include <iostream>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
using namespace std;
using namespace cv;

int main(int argc, char** argv) {
    Mat img = imread("binary image");
    Mat nonZeroCoordinates;
    findNonZero(img, nonZeroCoordinates);
    for (int i = 0; i < nonZeroCoordinates.total(); i++ ) {
        cout << "Zero#" << i << ": " << nonZeroCoordinates.at<Point>(i).x << ",
" << nonZeroCoordinates.at<Point>(i).y << endl;
    }
    return 0;
}
```

Method 2:

```
#include <iostream>
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>
using namespace std;
using namespace cv;

int main(int argc, char** argv) {
    Mat img = imread("binary image");
    for (int i = 0; i < img.cols; i++ ) {
        for (int j = 0; j < img.rows; j++) {
            if (img.at<uchar>(j, i) > 0) {
                cout << i << ", " << j << endl;     // Do your operations
            }
        }
    }
    return 0;
}
```

edited Sep 21 '17 at 10:45
alkasm 15.3k 1 51 69

answered Oct 8 '13 at 9:52
WangYudong 3,927 3 25 50

I get an error saying: 'findNonZero()' was not declared in this scope. What do I need to import to even be able to use this function? – DMor Oct 8 '13 at 9:56

Yes, I've successfully compiled and run everything from loading images, displaying windows to using Canny edge detection and thresholding. – DMor Oct 8 '13 at 10:46

Hope **Method 2** helps! – WangYudong Oct 8 '13 at 12:05

The answer to your problem is pretty simple, it needs at least OpenCV 2.4.4. – Ela782 Dec 11 '13 at 11:58

3 loop order should be reversed: iterate over rows, then columns. OpenCV's Mat container is stored in row-major order, so iterating over columns is cache unfriendly. – nicodjimenez Aug 7 '14 at 20:14

5

There is the following source code that was supplied for OpenCV 2.4.3, which may be helpful:

```
#include <opencv2/core/core.hpp>
#include <vector>

/*! @brief find non-zero elements in a Matrix
 *
 * Given a binary matrix (likely returned from a comparison
 * operation such as compare(), >, ==, etc, return all of
 * the non-zero indices as a std::vector<cv::Point> (x,y)
 *
 * This function aims to replicate the functionality of
 * Matlab's command of the same name
 *
 * Example:
 * \code
 *  // find the edges in an image
 *  Mat edges, thresh;
 *  sobel(image, edges);
 *  // theshold the edges
 *  thresh = edges > 0.1;
 *  // find the non-zero components so we can do something useful with them
later
 *  vector<Point> idx;
 *  find(thresh, idx);
 * \endcode
 *
 * @param binary the input image (type CV_8UC1)
 * @param idx the output vector of Points corresponding to non-zero indices in
the input
 */
void find(const cv::Mat& binary, std::vector<cv::Point> &idx) {

    assert(binary.cols > 0 && binary.rows > 0 && binary.channels() == 1 &&
binary.depth() == CV_8U);
    const int M = binary.rows;
    const int N = binary.cols;
    for (int m = 0; m < M; ++m) {
        const char* bin_ptr = binary.ptr<char>(m);
        for (int n = 0; n < N; ++n) {
            if (bin_ptr[n] > 0) idx.push_back(cv::Point(n,m));
        }
    }
}
```

Note - it looks like the function signature was wrong so I've changed the output `vector` to pass-by-reference.

edited Oct 8 '13 at 11:19

answered Oct 8 '13 at 9:23
Roger Rowland
23.9k 10 66 98

This function doesn't seem to help, my vector<Point coordinates; variable appears empty and won't even enter the loop. Here is my basic implementation. pastebin.com/Rznhb4wv – DMor Oct 8 '13 at 11:16

See above edit - it looks like the vector was being passed by value, so I've amended the code and added a note. – Roger Rowland Oct 8 '13 at 11:19

It still appears empty. I checked by using coordinates.size() and it equals 0. (variables explained in pastebin link above). – DMor Oct 8 '13 at 11:28

@Dmor574 Then perhaps you are not passing a single-channel greyscale image? The code above assumes CV_8UC1 and would need changing for anything else. If it's not that, then is it possible that your image really does have no zero pixels? The code is quite simple. – Roger Rowland Oct 8 '13 at 12:00

I believe that it might be possible that its in a different format, but @WangYudong, his second method worked (I'm unsure if its relevant). – DMor Oct 15 '13 at 3:02

-3

you can find it without using findNonZero() this opencv method. rather u can get it by simply using 2 for loops. here is the snippet. hope it can help u.

**

```
for(int i = 0 ;i <image.rows() ; i++){// image : the binary image
            for(int j = 0; j< image.cols() ; j++){
                double[] returned = image.get(i,j);
                int value = (int) returned[0];
                if(value==255){
                System.out.println("x: " +i + "\ty: "+j);// returned the (x,y)
//co ordinates of all white pixels.
                }
            }

        }
```

**

answered Jul 5 '16 at 5:25

Tanusree

9 3