

## What is gradient orientation and gradient magnitude

[Ask Question](#)

I am currently studying a module in computer vision called edge detection. I am trying to understand the meaning of gradient orientation and gradient magnitude.

[computer-vision](#)

asked Nov 6 '13 at 15:13



Jack Welch

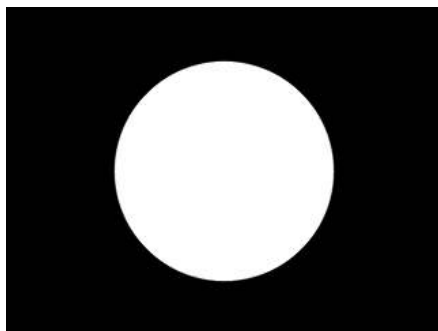
711 3 12 22

### 2 Answers

As explained by [Dima](#) in his [answer](#), you should be familiar with the mathematical concept of [gradient](#) in order to better understand the gradient in the field of image processing.

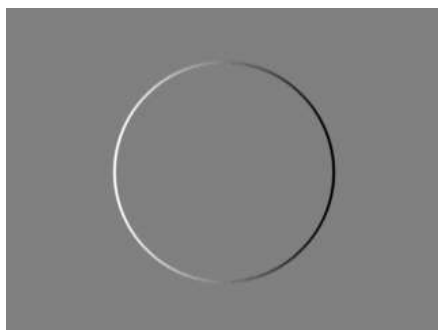
My answer is based on the [answer](#) of [mevatron](#) to this [question](#).

Here you find a simple initial image of a white disk on a black background:



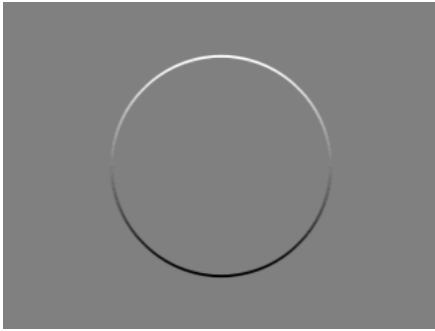
you can compute an approximation of the gradient of this image. As Dima explained in his answer, you have two components of the gradient, an horizontal and a vertical component.

The following images show you the horizontal component:



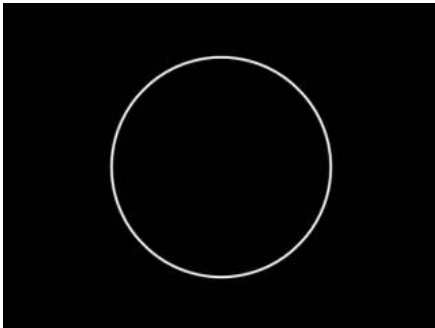
it shows how much the gray levels in your image change in the horizontal direction (it is the direction of positive  $x$ , scanning the image from left to right), this change is "encoded" in the gray level of the image of the horizontal component: the mean gray level means no change, the bright levels mean change from a dark value to a bright value, the dark levels mean a change from a bright value to a dark value. So, in the above image you see the brighter value in the left part of the circle because it is in the left part of the initial image that you have the black to white transition that gives you the left edge of the disk; similarly, in the above image you see the darker value in the right part of the circle because it is in the right part of the initial image that you have the white to black transition that gives you the right edge of the disk. In the above image, the inner part of the disk and the background are at a mean gray level because there is no change inside the disk and in the background.

We can make analogous observations for the vertical component, it shows how the image changes in the vertical direction, i.e. scanning the image from the top to the bottom.



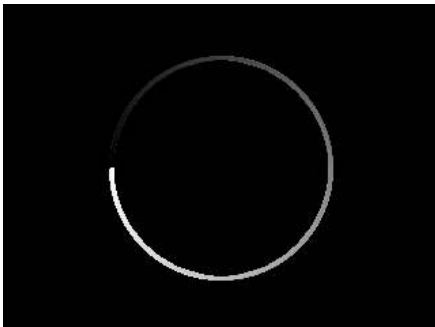
You can now combine the two components in order to get the magnitude of the gradient and the orientation of the gradient.

The following image is the magnitude of the gradient:

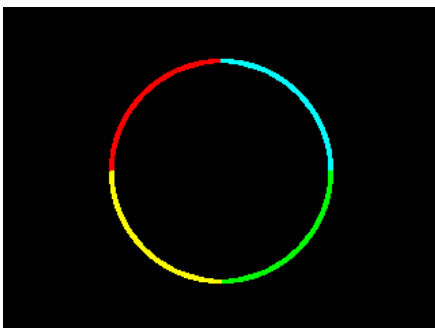


Again, in the above image the change in initial image is encoded in the gray level: here you see that white means an high change in the initial image while black means no change at all. So, when you look at the image of the magnitude of the gradient you can say "if the image is bright it means a big change in the initial image; if it is dark it means no change or very little change".

The following image is the orientation of the gradient:



In the above image the orientation is again encoded as gray levels: you can think at the orientation as the angle of an arrow pointing from the the dark part of the image to the bright part of the image; the angle is referred to an xy frame where the x runs from left to right while the y runs from top to bottom. In the above image you see all the grey level from the black (zero degree) to the white (360 degree). We can encode the information with color:



cyan: the angle is between 90 and 180 degree

green: the angle is between 180 and 270 degree

yellow: the angle is between 270 and 360 degree

Here it is the C++ OpenCV code for producing the above images.

**Pay attention to the fact that, for the computation of the orientation, I use the function `cv::phase` which, as explained in the [doc](#), gives an angle of 0 when both the vertical component and the horizontal component of the gradient are zero; that may be convenient but from a mathematical point of view is plainly wrong because when both components are zero the orientation is not defined and the only meaningful value for an orientation kept in a floating-point C++ type is a NaN.**

It is plainly wrong because a 0 degree orientation, for example, is already related to an horizontal edge and it cannot be used to represent something else like a region with no edges and so a region where orientation is meaningless.

```
// original code by https://stackoverflow.com/users/951860/mevatron
// see https://stackoverflow.com/a/11157426/15485
// https://stackoverflow.com/users/15485/uvts-cvs added the code for saving x and y
// gradient component

#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>

#include <iostream>
#include <vector>

using namespace cv;
using namespace std;

Mat mat2gray(const cv::Mat& src)
{
    Mat dst;
    normalize(src, dst, 0.0, 255.0, cv::NORM_MINMAX, CV_8U);

    return dst;
}

Mat orientationMap(const cv::Mat& mag, const cv::Mat& ori, double thresh = 1.0)
{
    Mat oriMap = Mat::zeros(ori.size(), CV_8UC3);
    Vec3b red(0, 0, 255);
    Vec3b cyan(255, 255, 0);
    Vec3b green(0, 255, 0);
    Vec3b yellow(0, 255, 255);
    for(int i = 0; i < mag.rows*mag.cols; i++)
    {
        float* magPixel = reinterpret_cast<float*>(mag.data + i*sizeof(float));
        if(*magPixel > thresh)
        {
            float* oriPixel = reinterpret_cast<float*>(ori.data + i*sizeof(float));
            Vec3b* mapPixel = reinterpret_cast<Vec3b*>(oriMap.data + i*3*sizeof(char));
            if(*oriPixel < 90.0)
                *mapPixel = red;
            else if(*oriPixel >= 90.0 && *oriPixel < 180.0)
                *mapPixel = cyan;
            else if(*oriPixel >= 180.0 && *oriPixel < 270.0)
                *mapPixel = green;
            else if(*oriPixel >= 270.0 && *oriPixel < 360.0)
                *mapPixel = yellow;
        }
    }

    return oriMap;
}

int main(int argc, char* argv[])
{
    Mat image = Mat::zeros(Size(320, 240), CV_8UC1);
    circle(image, Point(160, 120), 80, Scalar(255, 255, 255), -1, CV_AA);

    imshow("original", image);

    Mat Sx;
    Sobel(image, Sx, CV_32F, 1, 0, 3);

    Mat Sy;
    Sobel(image, Sy, CV_32F, 0, 1, 3);

    Mat mag, ori;
    magnitude(Sx, Sy, mag);
    phase(Sx, Sy, ori, true);
}
```

```
imwrite("hor.png",mat2gray(Sx));
imwrite("ver.png",mat2gray(Sy));

imshow("magnitude", mat2gray(mag));
imshow("orientation", mat2gray(ori));
imshow("orientation map", oriMap);
waitKey();


return 0;
}
```

edited May 23 '17 at 11:54

Community

11

answered Nov 9 '13 at 17:40

Alessandro Jacopson

9,824868116

5 Awesome answer. Keep it up – Vinoj John Hosan Jun 17 '15 at 13:05

But Is it possible to get gradient x and gradient y from magnitude of gradient? – Abc Aug 27 '16 at 7:08


@Abc Let's say  $x$  is the horizontal component of the gradient and  $y$  is the vertical at one specific pixel, now the magnitude of gradient  $M$  at that pixel is  $M=\sqrt{\text{pow}(x,2)+\text{pow}(y,2)}$  and so if you know just  $M$  it seems to me difficult to get  $x$  and  $y$  . – Alessandro Jacopson Aug 27 '16 at 9:23

I will be Lunatic if I don't vote for this answer. Just brilliant for beginners. – Whoami Jan 17 at 14:06

The gradient of a function of two variables  $x, y$  is a vector of the partial derivatives in the  $x$  and  $y$  direction. So if your function is  $f(x,y)$ , the gradient is the vector  $(f_x, f_y)$ . An image is a discrete function of  $(x,y)$ , so you can also talk about the gradient of an image.

The gradient of the image has two components: the  $x$ -derivative and the  $y$ -derivative. So, you can think of it as vectors  $(f_x, f_y)$  defined at each pixel. These vectors have a direction  $\text{atan}(f_y / f_x)$  and a magnitude  $\sqrt{f_x^2 + f_y^2}$ . So, you can represent the gradient of an image either an  $x$ -derivative image and a  $y$ -derivative image, or as direction image and a magnitude image.

answered Nov 6 '13 at 15:42

Dima

32.6k1258107