

# Zernike-Moments Based Edge Detection

## user manual

T. Osswald

Brussels  
March 26, 2021

### Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Sub-pixel Edge Detection . . . . .	1
1.2	Zernike Moments . . . . .	1
<b>2</b>	<b>Algorithm</b>	<b>3</b>
2.1	The code . . . . .	3
	<b>References</b>	<b>5</b>

# 1 Introduction

Edge detection has a wide range of applications. The edges on an object are generally what most people will draw when trying to represent it on paper, edges are generally unique to that given object and are therefore quite useful for recognizing it. For more academic purposes knowing the exact position of an edge will generally allow an experimentalist to know the position of the object, so all he needs is a camera and a computer to find the edge. This becomes especially useful for studying moving objects, where the lazy scientist doesn't want to go through the pain of checking every single frame of the camera.

## 1.1 Sub-pixel Edge Detection

Since digital cameras have a finite resolution, even with the sharpest and most precise adjustment of the lenses an edge will always be seen as a gradient over some pixels as shown in figure 3. To overcome this issue several methods have been devised to estimate where, in this gradient region, the real edge is located. The method first described by Ghosal and Mehrotra in [1] using Zernike moments is implemented in this code. It provides a mathematically sound description of what an edge is and detects it with high accuracy.

## 1.2 Zernike Moments

The usage of the Zernike polynomials for analysis of images is first described in [2] and its specific application to edge detection is detailed in [1], whose notation is followed in this manual. When written for the complex plane Zernike polynomials are as follows:

$$V_{nm}(\rho, \theta) = R_{nm}(\rho) \exp^{jm\theta}$$

$$R_{nm}(\rho) = \sum_{s=0}^{(n-|m|)/2} \frac{(-1)^s (n-s)! \rho^{n-2s}}{s! (\frac{n+|m|}{2} - s)! (\frac{n-|m|}{2} - s)!} \quad (1)$$

where  $n \geq 0$ ,  $n - |m|$  is an even positive integer and the coordinates  $\rho$  and  $\theta$  are valid only within the unit circle. The polynomials, being orthogonal with respect to each other, can serve as a new basis onto which the image with the intensity  $f(x, y)$  can be projected, similarly to what is done with the Fourier transform. The projection in this new space is given by the Zernike moments, whose discrete form, valid over the unit circle, is

$$A_{nm} = \sum_x \sum_y f(x, y) V_{nm}^*(\rho, \theta) \quad (2)$$

the  $*$  representing complex conjugation. The polynomials used for edge detection are  $V_{1,1}$  and  $V_{2,0}$  whose representation is shown in figure 1.

One useful feature of this new basis system is that, for a rotation of the original image by an angle  $\phi$ , the Zernike moments acquire only a change in their argument, while the

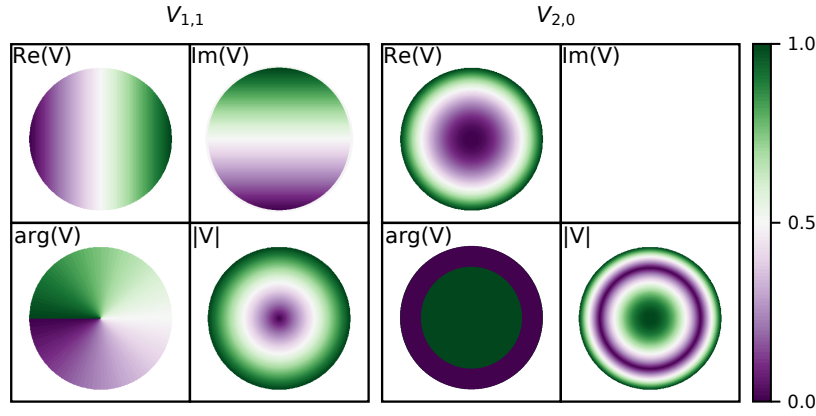


Figure 1: Relevant Zernike polynomials. All plots normalized.

magnitude remains the same. For the particular case of the relevant moments their rotated counterparts are

$$A'_{11} = A_{11} \exp^{j\phi} \quad \text{and} \quad A'_{20} = A_{20}. \quad (3)$$

If an ideal edge is subsequently defined with its rotation given by the angle  $\phi$ , the edge intensity by  $k$  and the distance from the center of the image to the edge by  $l$  as shown in figure 2; then it is possible to define this edge in a space of Zernike polynomials without having to go higher than  $n = 2$ .

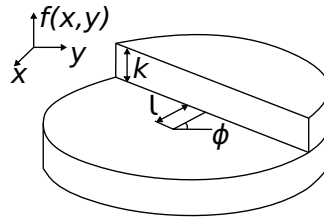


Figure 2: Edge model. Adapted from [1].

Furthermore each of the parameters can be written as a function of the Zernike moments:

$$\begin{aligned} l &= \frac{A_{20}}{A'_{11}} \\ k &= \frac{3A'_{11}}{2(1-l^2)^{3/2}} \end{aligned} \quad (4)$$

It is apparent that this method is using the whole image to detect the edge and finding it as a feature and not a gradient, as other algorithms would do. This gives it a higher robustness to noise and directly outputs the angle of the edge.

## 2 Algorithm

Before proceeding with the detection of the edges themselves some pre-treatment of the image is done to reduce noise and enhance the features of interest. For this reason a horizontal gaussian blurring (low-pass filter) was applied, which could be quite strong since there was little variation of the image over the horizontal coordinate as seen in figure 3. The level of the blurring should be adapted on a case by case scenario and for cases where the edge orientation is not known *a priori* it makes more sense to use a homogeneous blurring (as implemented in current version of code).

The calculation of Zernike moments of a given order can be efficiently implemented by previously defining  $V_{nm}^*(x', y')$  in a grid of points, more precisely a circular kernel. Its multiplication with  $f(x, y)$  and subsequent summation is inexpensive and can be repeated for a large set of images through equation 2. In what concerned a specific application the kernels used were of 5x5 pixels. Since the positions of the edge could change throughout the image, the Zernike moments were defined not only as one set for the image as a whole, but for a kernel around each pixel as seen in figure 3. The kernel was thus convolved over the whole image. Once one set of Zernike moments per pixel was obtained, the edge parameters ( $k$ ,  $l$  and  $\phi$ ) could be calculated.

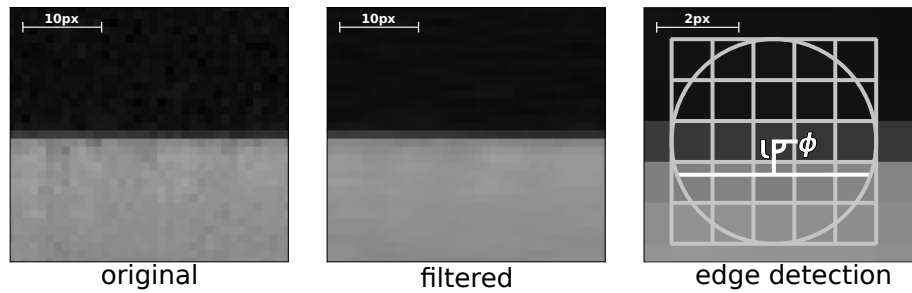


Figure 3: Processing of a section of the image. Low-pass filtering and detection of edge position with the kernel around a given pixel.

A lower and upper threshold was applied to the edge intensity  $k$  since a very low  $k$  indicates that there is no edge at all and a very high  $k$  can be caused by reflections or other artifacts of the image. To help reduce errors an upper limit was also set on  $l$ . This avoids having edges that are very close to the corner of the kernel which are more prone to errors. Finally  $\phi$  was thresholded to only allow horizontal edges, in cases where the edge orientation is not known then such a threshold should not be applied. With this type of filtering only the real edges of the image were detected. The calculated  $\phi$  and  $l$  could subsequently be used to determine the exact position of the edge.

### 2.1 The code

The code itself is commented and follows the same logic from before. The `main.py` file is the one that has to be run and will import the user defined functions in `functions.py` –

where the actual algorithm for the edge detection proper is defined under `ghosal_edge_v2` – and the external modules from `modules.py`. The functions are divided by "Tiers" where higher tier functions depend on lower tier ones. I tried to make it so that a user only has to edit the files inside the settings folder while leaving the code untouched.

- `dataFolder.csv`: contains the folder location where all the images are located
- `ghosal_edge_paramters.csv`: contains 7 parameters, more than one file of this type can be used, just make sure to have different names and to reference them correctly on `imageinfo.csv`
  1. kernel size
  2. minimum allowed value of  $k$
  3. maximum allowed value of  $k$
  4. maximum distance from center of kernel, where 1 is the kernel edge
  5. minimum allowed angle  $\phi$
  6. an obsolete parameter, ignore it
  7. the blurring strength
- `imageinfo.csv`: contains name of images, weather they should be processed ("x") or no, and the file containing the edge parameters to be used.
- `saveTo.csv`: where should the txt results be saved

Feel free to contact the author for any clarification.

---

## References

- [1] Sugata Ghosal and Rajiv Mehrotra. Orthogonal moment operators for subpixel edge detection. *Pattern recognition*, 26(2):295–306, 1993.
- [2] Michael Reed Teague. Image analysis via the general theory of moments. *JOSA*, 70(8):920–930, 1980.

T. Osswald  
Porto, Portugal  
*E-mail address*, [tobiasosswald@gmail.com](mailto:tobiasosswald@gmail.com)