Eötvös Loránd University
Faculty of Informatics
Department of Numerical Analysis

Color image analysis and recognition using Zernike moments

TDK thesis

Supervisor:
Zsolt Németh
Title title title

Author:
Gergely Nagy
Computer Science MSc
1st year

Budapest

Abstract

Image moments and moment invariant features are widely used for image analysis and pattern recognition. The system of orthogonal Zernike polynomials (defined over the complex unit disk) proved to be a useful basis for series expansions because of certain invariance properties.

Conventionally, for multichannel color images, RGB decomposition or grayscale conversion was used. However, in the recent past, quaternion algebra has been employed to various conventional moments to analyze a color image holistically. Guo and Zhu [1] introduced quaternion Fourier-Mellin moments (QFMMs) which are an extension of the conventional Fourier-Mellin moments for the grayscale image. They also proposed their invariants on rotation, scale, and translation for color object recognition. Chen et al. [2] proposed the quaternion Zernike moments (QZMs), generally overperforming other similar approaches in these aspects, due to the natural invariances of Zernike functions. The same quaternion techniques were applied successfully to other function systems (e.g. [3, 4]), yielding similar results.

In this thesis we introduce a method of transforming a digital RGB image inside the unit circle onto a points system providing discrete orthogonality. Using this points system for the discretization of the QZMIs, we have achieved significant improvements in the image recognition and reconstruction capabilities of the method, especially under noisy conditions.

Contents

Chapter 1

Introduction

TODO Intro

Something about the topic, why is it relevant, important...

1.1 Contributions

What have we achieved, what is new?

1.2 Structure of the paper

Structure

Chapter 2

Background

This chapter contains a summary and overview of the concepts used in this thesis and previous results this work is based on, such as image moments and their relevance in image analysis, and more specifically Zernike moments and the state-of-the-art of their application for both grayscale and color image analysis. Furthermore, some examples are provided to show the relevance and use cases of such image moments.

2.1 Image moments

In general, image moments are certain descriptive values calculated using the pixel intensities of an image. Different moments can be used to extract certain properties from a picture, for example the centroid of a grayscale image can be calculated as

$$\{\overline{x}, \overline{y}\} = \left\{ \frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}} \right\}$$

where M_{ij} are the regular image moments defined as

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y) \tag{2.1}$$

with $I(x, y)$ being the pixel value at the coordinates (x, y).

Moment invariants

Using the image moments, moment invariants can be defined, which are invariant to certain transformations, such as rotation, scaling, and translation.

These moment invariants are widely used in applications for pattern matching and image recognition [5, 6, 7]. In particular, moment invariants can be used in medical applications, such as solving the Pathological Brain Detection problem [8].

2.2 Zernike moments

Zernike polynomials, first introduce by Zernike in 1934 [9] are a system of complex, orthogonal polynomials defined on the unit disc. In polar coordinates the Zernike polynomial $V_{n,m}$ ($n \in \mathbb{N}$, $m \in \mathbb{Z}$, $n \geq |m|$ and $n - |m|$ is even) is defined as

$$V_{n,m}(r, \theta) = R_{n,m}(r)e^{im\theta}$$

where $R_{n,m}(r)$ are the radial polynomials defined as

$$R_{n,m}(r) = \sum_{k=0}^{\frac{n-|m|}{2}} \frac{(-1)^k (n-k)!}{k! \left(\frac{n+|m|}{2} - k\right)! \left(\frac{n-|m|}{2} - k\right)!} r^{n-2k} \tag{2.2}$$

Zernike moments are image moments, defined for grayscale images inside the unit circle, using the Zernike polynomials. The system of Zernike polynomials proved to be a suitable basis for series expansions, as moment invariants could easily be constructed using the Zernike moments [10].

The Zernike moment of order n and repetition m of a grayscale, continuous image function $f(r, \theta)$ given in polar coordinates, is defined as

$$Z_{n,m}(f) = \frac{n+1}{\pi} \int_0^1 \int_0^{2\pi} f(r, \theta) V_{n,m}^*(r, \theta) r \, \mathrm{d}r \, \mathrm{d}\theta$$

Because the Zernike polynomials are orthogonal, the following reconstruction of the image function is possible, using Zernike moments only up to a finite M degree:

$$f(r, \theta) \approx \sum_{n=0}^{M} \sum_{m=-n}^{n} Z_{n,m}(f) V_{n,m}(r, \theta)$$

Since digital images are not represented in polar coordinates and are not defined only over the unit disc, a transformation of the image onto the unit disk is needed. The most commonly used transformation is a linear transformation from the image coordinates to

a suitable square inside the unit circle. This transformation is described in more detail in Section 2.4. After the linear transformation, the following discrete approximation can be used to calculate the Zernike moments of a digital image $f(x, y)$

$$Z_{n,m}(f) = \frac{2(n+1)}{\pi(N-1)^2} \sum_{x=1}^{N} \sum_{y=1}^{N} f(x,y) V^*_{n,m}(r_{x,y}, \theta_{x,y})$$

where N is the size of the image, and $(r_{x,y}, \theta_{x,y})$ are the polar coordinates corresponding to the (x, y) image coordinates.

2.3 Quaternions and color image moments

The previously defined Zernike moments can only be used for grayscale images, but nowadays most images are RGB, so a method is needed to use image moments for color images.

Conventionally, for multichannel, color images, two main approaches can be used. Either the image is converted to grayscale so that the moments defined for grayscale images could be used or, after RGB-decomposition, the grayscale method is used on each channel of the image [11].

More recently, the algebra of quaternions was employed to various conventional moments so that color images can be analysed holistically.

Quaternions are a generalization of complex numbers, consisting of one real and three imaginary parts. A quaternion q can be represented in the form $q = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$, where $a, b, c, d \in \mathbb{R}$ and $\mathbf{i}, \mathbf{j}, \mathbf{k}$ are the imaginary units, defined by the multiplication rules $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$. The set of quaternions is denoted by $\mathbb{H}$.

For example, Guo and Zhu [1] introduced quaternion Fourier-Mellin moments (QFMMs) which are an extension of the conventional Fourier-Mellin moments. Similarly, the same quaternion techniques were applied successfully to other function systems (e.g. [3, 4]), yielding similar results.

Chen et al. [12, 2] proposed the quaternion Zernike moments (QZMs), the extension of conventional Zernike moments to color images using quaternions. Generally, this method overperforms other similar approaches in color image recognition, due to the natural invariances of Zernike functions.

The main idea behind these extensions is that an RGB image can be viewed as a pure quaternion valued function, with each color component corresponding to one of the imaginary units.

Quaternion Zernike moments

Let $f(r, \theta)$ be a pure quaternion valued, continuous RGB image function, defined in polar coordinates on the unit circle. Each color component corresponds to one of the imaginary units. Let $\boldsymbol{\mu} = \frac{\mathbf{i}+\mathbf{j}+\mathbf{k}}{\sqrt{3}}$ be a unit pure quaternion.

Since the multiplication of quaternions is not commutative, right-side and left-side quaternion Zernike moments can also be defined. The right-side QZM of order n and repetition m is defined as

$$Z^R_{n,m}(f) = \frac{n+1}{\pi} \int_0^1 \int_0^{2\pi} R_{n,m}(r) f(r,\theta) e^{-\boldsymbol{\mu} m\theta} r \,\mathrm{d}r \,\mathrm{d}\theta,$$
$$n \geq |m| \text{ and } n - |m| \text{ is even}$$

The left-side QZMs are defined as

$$Z^L_{n,m}(f) = \frac{n+1}{\pi} \int_0^1 \int_0^{2\pi} R_{n,m}(r) e^{-\boldsymbol{\mu} m\theta} f(r,\theta) r \,\mathrm{d}r \,\mathrm{d}\theta$$

The main difference, compared to the conventional Zernike moments is that instead of the complex-valued Zernike polynomials, QZMs use the quaternion valued generalization of the Zernike polynomials as a basis for the series expansion.

Similarly to non-quaternion Zernike moments, the original image can be approximated by using either right-side or left-side QZMs only up to a finite M degree.

$$f(r,\theta) \approx \sum_{n=0}^{M} \sum_{m=-n}^{n} Z^R_{n,m}(f) R_{n,m}(r) e^{\boldsymbol{\mu} m\theta}$$
$$f(r,\theta) \approx \sum_{n=0}^{M} \sum_{m=-n}^{n} e^{\boldsymbol{\mu} m\theta} Z^L_{n,m}(f) R_{n,m}(r)$$

In this thesis, only the right-side QZMs will be used, because of the following relation between right- and left-side QZMs:

$$Z^L_{n,m}(f) = -(Z^R_{n,m}(f))^*$$

The discretization of the QZMs and the transformation of arbitrary digital RGB images onto the unit disk is described in detail in Section 2.4

2.4 Discretization of QZMs

The conventional Zernike moments and the QZMs are defined in terms of polar coordinates, for image functions, whose domain is the unit disk. On the other hand, digital images are usually defined in image coordinates, with the coordinates being integers ranging from 0 to $N-1$ (the number of pixels along each axis).

There are two common ways to linearly transform a square image from image coordinates to polar coordinates inside the unit circle, using only translation and scaling [13].

The first method is to transform the entire image inside the unit circle, as shown on Figure 2.1. This way all of the pixels will be used for calculating the (quaternion) Zernike moments, but some areas of the unit disk will remain empty, no pixels fall on those areas.

The other method is to transform the image such that the unit disk becomes the inscribed circle of the square image, as shown on Figure 2.2. This method fills the entire unit disk with pixels, but some pixels fall outside the unit cirle and thus will not be used for the calculation of (quaternion) Zernike moments.

The polar coordinates (r, θ) corresponding to the image coordinates (x, y) can be calculated by the following formulas:

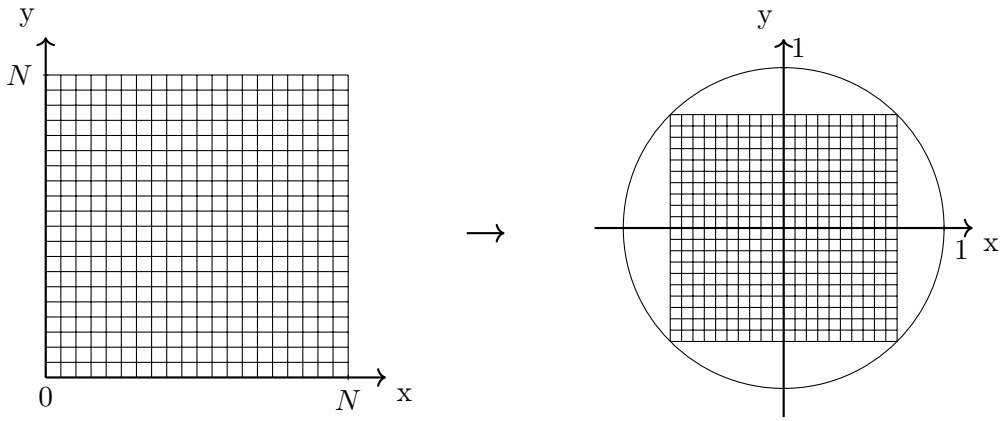
$$r = \sqrt{(c_1 x + c_2)^2 + (c_1 y + c_2)^2}$$
$$\theta = \tan^{-1}\left(\frac{c_1 y + c_2}{c_1 x + c_2}\right)$$

where c_1, $c_2 \in \mathbb{R}$ depend on which of the previously mentioned transformations is used. A $\lambda \in \mathbb{R}$ scaling factor is also defined for each transformation. For the first one (Figure 2.1): $c_1 = \frac{\sqrt{2}}{N-1}$, $c_2 = -\frac{1}{\sqrt{2}}$, and $\lambda = \frac{2}{\pi}$. While for the other transformation (Figure 2.2): $c_1 = \frac{2}{N-1}$, $c_2 = -1$ and $\lambda = 1$.

Using either one of the previously defined transformations, a points system on the unit disk is obtained, and for each point exactly one pixel value is assigned. When this points system is used for the discretization of QZMs, the following formula can be used to approximate the QZMs of the original image function:

$$Z^R_{n,m}(f) = \lambda \frac{n+1}{(N-1)^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} R_{n,m}(r_{x,y}) f(x,y) e^{-\boldsymbol{\mu} m \theta_{x,y}}$$

where $f \in \mathbb{R}^2 \to \mathbb{H}$ is the RGB image defined in image coordinates, $(r_{x,y}, \theta_{x,y})$ are the polar coordinates belonging to the image coordinates (x, y), and λ is the previously defined scaling factor.



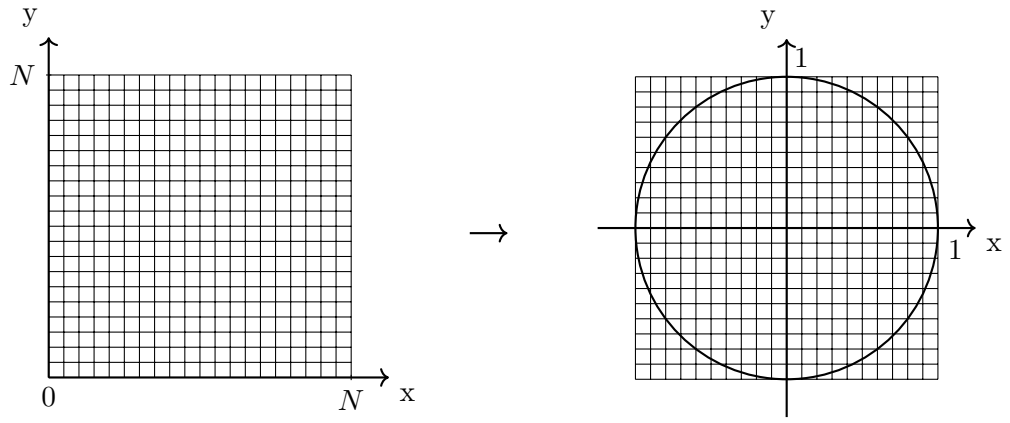
(a) An $N \times N$ image on the image plane (b) The image transformed inside the unit circle

Figure 2.1: The image before and after applying the transformation inside the unit disk

In the rest of this thesis, for comparison purposes, the transformation of the image inside the unit circle will be used as Chen et al. [2] used this discretization to present their results.

Discrete orthogonality

Even though the Zernike polynomials are a set of orthogonal polynomials on the complex unit disk, the previously defined points systems do not provide discrete orthogonality. This lack of discrete orthogonality means that the discretization of the series expansion based on the Zernike polynomials will introduce some errors and redundancy, thus decreasing the image reconstruction and pattern recognition capabilities of the method.



(a) An $N \times N$ image on the image plane (b) The image transformed onto the unit disk

Figure 2.2: The image before and after applying the transformation onto the unit disk

2.5 Quaternion Zernike Moment Invariants (QZMIs)

Chen et al. [12, 2] proposed the constructions described in this section in order to create rotation, scaling and translation invariant moment invariants using the quaternion Zernike moments.

Translation invariance

In order to achieve translation invariance, the common centroid of all three color channels can be calculated by using zero-order and first-order geometric moments, as described by Suk and Flusser [11]. If the origin of the polar coordinate system is placed on this common centroid (x_c, y_c), then the QZMs calculated in this coordinate system will be translation invariant. Let $\overline{Z}^R_{n,m}(f)$ denote the translation invariant QZMs.

Rotation invariance

To achieve rotation invariance consider the image rotated by some degree α: $f'(r,\theta) = f(r,\theta-\alpha)$. Chen et al. [2] proved that for the rotated image $Z^R_{n,m}(f') = Z^R_{n,m}(f)e^{-\boldsymbol{\mu} m\theta}$ and $Z^L_{n,m}(f') = e^{-\boldsymbol{\mu} m\theta}Z^L_{n,m}(f)$. Because of the these properties

$$\Phi^m_{n,k} = Z^R_{n,m}(f)Z^L_{k,-m}(f) = -Z^R_{n,m}(f)(Z^R_{k,m}(f))^*$$

is invariant to rotation. In the following chapters, this will be referred to as a quaternon Zernike moment rotation invariant (QZMRI).

Previously, the modulus $|Z^R_{n,m}(f)|$ was used to achieve rotation invariance [12], but this provided only one real-valued invariant, whereas the following construction provides a quaternion-values invariant.

Scaling invariance

For non-negative integers m and l, Chen et al. [2] constructed the following scaling invariants utilizing the symmetric property of the radial polynomials with respect to m and an alternate form of the QZMs.

$$c^{t,k}_{m,l} = (-1)^{l-k} \frac{(m+2l+1)(m+k+l)!}{(l-k)!(k-t)!(m+k+t+1)!}$$

$$L^R_{m+2l,m}(f) = \sum_{t=0}^{l} \sum_{k=t}^{l} \left(\sqrt{|Z^R_{0,0}(f)|} \right)^{-(m+2k+2)} c^{t,k}_{m,l} Z^R_{m+2t,m}(f)$$

Combined RST invariance

Similarly to using QZMs to define rotation invariants, the previously defined scaling invariants can also be used to construct $\varphi^m_{n,k} = L^R_{n,m}(f)(L^R_{k,m}(f))^*$, which is invariant to rotation and scaling.

In order to achieve translation invariance, throughout the construction of the scaling invariants $L^R_{n,m}$ the translation invariant QZMs ($\overline{Z}^R_{n,m}(f)$) can be used, thus defining the $\overline{L}^R_{n,m}$ translation and scaling invariants.

Furthermore, $\overline{\varphi}^m_{n,k} = \overline{L}^R_{n,m}(f)(\overline{L}^R_{k,m}(f))^*$ is invariant to rotation, scaling and translation [2]. This is called a quaternion Zernike moment invariant (QZMI).

2.6 Applications

Invarianst ad vissza, pl neural networkbe, gepi tanulasba jo

Chapter 3

Math?

3.1 Dummy

Chapter 4

Implementation

This chapter presents the tools and methods used for implementing a program for calculating the QZMIs of an image using both methods of discretization.

4.1 Programming language and libraries used

The implementation was created using the Python programming language [14] and relying on the Numpy [15] and Numba [16] libraries to achieve efficient and fast computation of the moment invariants, as well as the Python Imaging Library (PIL, Pillow) [17] for image manipulation.

Numpy provided a way to efficiently work with arrays and matrices. It also provided the quaternion package, which supports the quaternion data type.

Using the just-in-time compilation (JIT) capabilities of Numba (i.e. the `@jit` annotation), the computationally heavy parts of the implementation could be made almost as fast as native code. The disadvantage of using JIT is that it limits the available types and functions, so in the implementation the use of `@jit` is kept only to the critical, computationally intensive functions.

4.2 Calculating moments and moment invariants

To obtain the quaternion Zernike moment invariants (QZMIs) of an image as described in Section 2.5, the quaternion Zernike moments (QZMs) must be calculated first. Chen

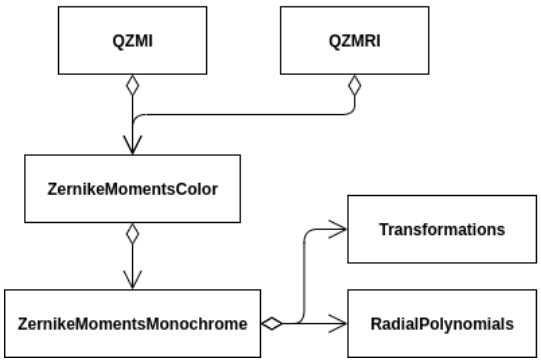


Figure 4.1: The relationships between the classes used to calculate QZMIs and QZMRIs

et al. [2] showed that instead of calculating the QZMs directly by using the algebra of quaternions, it is possible to calculate the real and the imaginary parts of the quaternion-valued QZM individually by using some linear combination of the real and imaginary parts of the complex-valued, single channel Zernike moments. This means that the single channel Zernike moments have to be calculated for all three of the RGB color channels. Furthermore, the calculation of the Zernike moments requires the computation of the radial polynomials, introduced in (2.2).

To calculate the required values, the following four classes were created, each relying on the next one to perform the computation.

- `QZMI`, for calculating the quaternion Zernike moment invariants
- `ZernikeMomentsColor`, for calculating the quaternion-valued Zernike moments
- `ZernikeMomentsMonochrome`, for calculating the complex-valued, single channel Zernike moments
- `RadialPolynomials`, for calculating the values of the radial polynomials at a given point

The relationships between these classes are shown on Figure 4.1, as well as the `QZMRI` class, which calculates the rotation invariant QZMs needed for some test cases. A description of the algorithms used in these classes and the data stored by them is given below.

Since the work presented in this thesis involves changing the way an image is transformed from image coordinates to polar coordinates inside the unit circle, the classes were made modular with respect to the transformation used. This makes it easy to create and

```
@jit(void(float64, int32, float64[:,:]), nopython=True)
def calculateRadialPolynomials(r, P, values):
  values[0,0] = 1
  values[1,1] = r
  for n in range(2,P + 1):
    h = n*(n - 1)*(n - 2)
    K2 = 2*h
    values[n,n] = (r**n)
    values[n,n-2] = n*values[n,n] - (n-1)*values[n-2,n-2]
    for m in range(n-4,-1,-2):
      K1 = (n + m)*(n - m)*(n - 2)/2
      K3 = (-1)*m*m*(n - 1) - h
      K4 = (-1)*n*(n + m - 2)*(n - m - 2)/2
      r2 = r**2
      values[n,m] = ((K2*r2+K3)*values[n-2,m]+K4*values[n-4,m])/K1
```

Figure 4.2: Function for calculating radial polynomial values

test a new image transformation function with the interface expected by the calculating classes.

Radial polynomials

To calculate the value of all $R_{n,m}$ radial polynomials up to some maximal degree P at a point $r \in [0, 1]$, the modified Kintner's method was used, as described in [13]. This algorithm computes the value of $R_{n,m}(r)$ for all $0 \leq |m| \leq n \leq P$, $(n - |m|$ is even) with complexity $\mathcal{O}(N^2)$. This method is ideal for the computation of all radial polynomial values up to a maximum degree.

Since this method is called many times during the calculation of Zernike moments, just-in-time compilation was used to further increase efficiency. Figure 4.2 shows the JIT-enabled function.

Complex Zernike moments

The `ZernikeMomentsMonochrome` class calculates the conventional Zernike moments of degree at most P of a square $N \times N$, single channel (grayscale) image. The algorithm is based directly on the discretized definition of the Zernike moments. For the original

linear transformation of the image onto the unit disk, this gives:

$$\begin{aligned} Z_{n,m}(f) &= \lambda \frac{(n+1)}{(N-1)^2} \sum_{x=1}^{N} \sum_{y=1}^{N} f(x,y) V^*_{n,m}(r_{x,y}, \theta_{x,y}) \\ &= \lambda \frac{(n+1)}{(N-1)^2} \sum_{x=1}^{N} \sum_{y=1}^{N} f(x,y) R_{n,m}(r_{x,y}) e^{-\boldsymbol{i} m \theta_{x,y}} \\ &= \lambda \frac{(n+1)}{(N-1)^2} \sum_{x=1}^{N} \sum_{y=1}^{N} f(x,y) R_{n,m}(r_{x,y}) (\cos(m\theta_{x,y}) - \boldsymbol{i} \sin(m\theta_{x,y})) \end{aligned}$$

where $0 \leq |m| \leq n \leq P$, $n - |m|$ is even, $(r_{x,y}, \theta_{x,y})$ are the (x, y) coordinates transformed to the unit disk, λ is the scaling parameter also given by the transformation (as described in Section 2.4) and f is the real-valued, grayscale image.

By precomputing the sin and cos values for all possible m and $\theta_{x,y}$ values, as well as the values of the radial polynomials, this formula gives an efficient way of calculating the real and imaginary parts of the Zernike moments separately. This way only primitive data types have to be used during the computation and it can be made more efficient using JIT.

Also, there is no need to calculate the Zernike moments for negative m values, as the $Z_{n,m}(f) = Z_{n,-m}(f)^*$ identity can be used later to obtain those values.

Quaternion Zernike moments

The class `ZernikeMomentsColor` calculates the quaternion Zernike moments of an RGB image. First, the conventional Zernike moments for each of the three color channels are calculated, then the relationship between QZMs and Zernike moments is used to construct the quaternions [2].

$$\begin{aligned} Z^R_{n,m}(f) = &- \frac{1}{\sqrt{3}} \left(Im(Z_{n,m}(f_R)) + Im(Z_{n,m}(f_G)) + Im(Z_{n,m}(f_B)) \right) \\ &+ \left[Re(Z_{n,m}(f_R)) + \frac{1}{\sqrt{3}} \left(Im(Z_{n,m}(f_G)) - Im(Z_{n,m}(f_B)) \right) \right] \boldsymbol{i} \\ &+ \left[Re(Z_{n,m}(f_G)) + \frac{1}{\sqrt{3}} \left(Im(Z_{n,m}(f_B)) - Im(Z_{n,m}(f_R)) \right) \right] \boldsymbol{j} \\ &+ \left[Re(Z_{n,m}(f_B)) + \frac{1}{\sqrt{3}} \left(Im(Z_{n,m}(f_R)) - Im(Z_{n,m}(f_G)) \right) \right] \boldsymbol{k} \end{aligned}$$

where f is an RGB image and f_R, f_G, f_B are the red, green and blue color channels respectively.

Again, only the QZMs $Z^R_{n,m}$ $(m \geq 0)$ are calculated, because for $m < 0$ the $Z^R_{n,m}(f) = Z^R_{n,-m}(f)^*$ equality can be used.

Invariants

The class `QZMI` is responsible for computing the combined rotation, scaling and translation (RST) invariant moments, while the class `QZMRI` computes the moments which are invariant only to rotation. The invariants for scaling and rotation are calculated directly using the QZMs, based on the formulas described in Section 2.5.

To achieve translation invariance, the common centroid of the RGB image is calculated based on the formulas described by Suk and Flusser [11].

$$\{x_c, y_c\} = \left\{ \frac{M_{10}(f_R) + M_{10}(f_G) + M_{10}(f_B)}{M_{00}(f_R) + M_{00}(f_G) + M_{00}(f_B)}, \frac{M_{01}(f_R) + M_{01}(f_G) + M_{01}(f_B)}{M_{00}(f_R) + M_{00}(f_G) + M_{00}(f_B)} \right\}$$

where f_R, f_G, f_B are the grayscale images corresponding to the red, green and blue color channels respectively, and M_{10}, M_{01}, M_{00} are the regular image moments introduced in (2.1). The original image f is then translated in image coordinates such that the origin falls on the common centroid $\{x_c, y_c\}$, and the other invariants are then calculated based on this translated image.

4.3 New image transformation

The new image transformation, as described in Section 3.1, requires the calculation of the roots of the n^{th} degree Legendre polynomials P_n, as well as calculating the integrals of the Lagrange interpolating polynomials over the roots of P_n. Furthermore, since when applying the linear transformation from image coordinates to polar coordinates, the pixels of the image do not fall exactly on any point in the new discrete points system, other interpolating methods have to be used to approximate the image values at these points.

Because of the modularity of the previously described classes, it is possible to swap the old transformation to the new one. With some minor modifications during the calculation of the conventional Zernike moments because of the new discretization formula containing a different measure, the previous classes can be used to obtain the QZMIs using the new method of discretization.

Roots of Legendre polynomials

The roots of the Legendre polynomial P_n are essential for the calculation of the new points system. An explicit formula, which gives the roots does not exist, thus an efficient and fast iterative algorithm is needed to calculate these roots.

The n^{th} degree Legendre polynomial P_n satisfies the following differential equation:

$$(1 - x^2)P_n''(x) - 2xP_n'(x) + n(n+1)P_n(x) = 0 \tag{4.1}$$

A fast algorithm for calculating the roots of P_n, based on this differential equation was presented by Glaser et al. [18]. The algorithm uses a second-order Runge-Kutta method (namely the midpoint method) to solve the Prüfer-transformed version of (4.1) for some given initial condition. A first approximation for a root of P_n can be obtained from the solution of the initial value problem. This approximation is then further refined by Newton's method. Subsequent roots can be calculated using the same method but starting from different initial conditions defined by the previous root.

In practice, this algorithm calculates the roots of P_n with accuracy up to machine precision in only a fix, limited number of iterations for both the Runge-Kutta and the Newton's method.

Gaussian quadratures

4.4 Tests

Methods for calculating roots of Legendre poly

Quadratures

Program

Chapter 5

Tests/improvements

In this chapter we present the different tests that were performed to compare the capabilities of the old and new methods.

In total, four kinds of tests were conducted:

- Invariance
- Image reconstruction
- Image recognition
- Template matching

All of these tests were run for both the old and new methods and the results have been compared.

5.1 Invariance

5.2 Image reconstruction

5.3 Image recognition

5.4 Template matching

Chapter 6

Conclusion

TODO conclusion

Future work, hogyan tovabb ... (akar sajat fejezetebe is)

Acknowledgment

The project was made possible by the support of the European Union and the cofunding of the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00001)

Bibliography

[1] Li-Qiang Guo and Ming Zhu. Quaternion fourier–mellin moments for color images. *Pattern Recognition*, 44(2):187 – 195, 2011. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2010.08.017. URL `http://www.sciencedirect.com/science/article/pii/S0031320310004036`.

[2] Beijing Chen, Huazhong Shu, Hui Zhang, Gang Chen, Christine Toumoulin, Jean-Louis Dillenseger, and Limin Luo. Quaternion zernike moments and their invariants for color image analysis and object recognition. *Signal Processing*, 92:308–318, 02 2012. doi: 10.1016/j.sigpro.2011.07.018.

[3] Zhuhong Shao, Huazhong Shu, Jiasong Wu, Beijing Chen, and Jean Louis Coatrieux. Quaternion bessel–fourier moments and their invariant descriptors for object reconstruction and recognition. *Pattern Recognition*, 47(2):603 – 611, 2014. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2013.08.016. URL `http://www.sciencedirect.com/science/article/pii/S0031320313003373`.

[4] Hongqing Zhu, Yan Yang, Zhiguo Gui, Yu Zhu, and Zhihua Chen. Image analysis by generalized chebyshev–fourier and generalized pseudo-jacobi–fourier moments. *Pattern Recognition*, 51:1 – 11, 2016. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2015.09.018. URL `http://www.sciencedirect.com/science/article/pii/S0031320315003490`.

[5] Feng Zhang, Shang-qian Liu, Da-bao Wang, and Wei Guan. Aircraft recognition in infrared image using wavelet moment invariants. *Image and Vision Computing*, 27:313–318, 03 2009. doi: 10.1016/j.imavis.2008.08.007.

[6] Yi Hsien Lin and Chin-Hsing Chen. Template matching using the parametric template vector with translation, rotation and scale invariance. *Pattern Recognition*, 41(7):2413–2421, July 2008. ISSN 0031-3203. doi: 10.1016/j.patcog.2008.01.017.

[7] Florica Mindru, Tinne Tuytelaars, Luc Van Gool, and Theo Moons. Moment invariants for recognition under changing viewpoint and illumination. *Computer Vision and Image Understanding*, 94(1):3 – 27, 2004. ISSN 1077-3142. doi: https://doi.org/10.1016/j.cviu.2003.10.011. URL `http://www.sciencedirect.com/science/`

article/pii/S1077314203001930. Special Issue: Colour for Image Indexing and Retrieval.

[8] Yu-Dong Zhang, Shui-Hua Wang, Xiao-Jun Yang, Zheng-Chao Dong, Ge Liu, Preetha Phillips, and Ti-Fei Yuan. Pathological brain detection in mri scanning by wavelet packet tsallis entropy and fuzzy support vector machine. *SpringerPlus*, 4:716, 2015. ISSN 2193-1801. doi: 10.1186/s40064-015-1523-4. URL https://europepmc.org/articles/PMC4656268.

[9] von F. Zernike. Beugungstheorie des schneidenver-fahrens und seiner verbesserten form, der phasenkontrastmethode. *Physica*, 1(7):689 – 704, 1934. ISSN 0031-8914. doi: https://doi.org/10.1016/S0031-8914(34)80259-5. URL http://www.sciencedirect.com/science/article/pii/S0031891434802595.

[10] A. Khotanzad and Y. H. Hong. Invariant image recognition by zernike moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5):489–497, 1990.

[11] Tomáš Suk and Jan Flusser. Affine moment invariants of color images. In Xiaoyi Jiang and Nicolai Petkov, editors, *Computer Analysis of Images and Patterns*, pages 334–341, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-03767-2.

[12] B. Chen, H. Shu, H. Zhang, G. Chen, and L. Luo. Color image analysis by quaternion zernike moments. In *2010 20th International Conference on Pattern Recognition*, pages 625–628, 2010.

[13] Chee-Way Chong, P. Raveendran, and R. Mukundan. A comparative analysis of algorithms for fast computation of zernike moments. *Pattern Recognition*, 36(3):731 – 742, 2003. ISSN 0031-3203. doi: https://doi.org/10.1016/S0031-3203(02)00091-2. URL http://www.sciencedirect.com/science/article/pii/S0031320302000912.

[14] Python Software Foundation. The python language reference. https://docs.python.org/3/reference/, 2020. [Accessed: 2020.05.06.].

[15] The SciPy community. Numpy v1.18 manual. https://numpy.org/doc/stable/, 2020. [Accessed: 2020.05.06.].

[16] Anaconda Inc. et al. Numba reference manual. http://numba.pydata.org/numba-doc/latest/reference/index.html, 2020. [Accessed: 2020.05.06.].

[17] Alex Clark et al. Pillow reference manual. https://pillow.readthedocs.io/en/stable/, 2020. [Accessed: 2020.05.06.].

[18] Andreas Glaser, Xiangtao Liu, and Vladimir Rokhlin. A fast algorithm for the calculation of the roots of special functions. *SIAM J. Scientific Computing*, 29: 1420–1438, 01 2007. doi: 10.1137/06067016X.