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.

Quaternion representation of color images

A quaternion, q, was defined by Hamilton [12] as a generalization of the complex numbers:

$$q = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$$

The real number a, b, c and d are called the components of q , and the imaginary units i, j and k are defined according to the following rules:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1,$$
$$\mathbf{ij} = -\mathbf{ji} = \mathbf{k},\ \mathbf{jk} = -\mathbf{kj} = \mathbf{i},\ \mathbf{ki} = -\mathbf{ik} = \mathbf{j}.$$

Therefore, the set of quaternions $\mathbb{H}$ is an *algebra*, where a quaternion is called pure quaternion when $a = 0$. The conjugate and modulus of a quaternion are respectively defined by

$$q^* = a - b\mathbf{i} - c\mathbf{j} - d\mathbf{k},$$
$$|q| = \sqrt{a^2 + b^2 + c^2 + d^2}.$$

Ell and Sangwine [13] utilized quaternions to represent a color image, $f : \mathbb{R}^2 \to \mathbb{R}^3$, as follows:

$$f(x, y) = \mathbf{i} f_R(x, y) + \mathbf{j} f_G(x, y) + \mathbf{k} f_B(x, y),$$

where functions $f_R, f_G, f_B : \mathbb{R}^2 \to \mathbb{R}$ represent the red, green and blue components of the (x, y) pixel, respectively.

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.

Quaternion Zernike moments

Chen et al. [14, 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.

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.

$$\begin{aligned} f(r,\theta) &\approx \sum_{n=0}^{M} \sum_{m=-n}^{n} Z_{n,m}^{R}(f) R_{n,m}(r) e^{\boldsymbol{\mu} m\theta} \\ f(r,\theta) &\approx \sum_{n=0}^{M} \sum_{m=-n}^{n} e^{\boldsymbol{\mu} m\theta} Z_{n,m}^{L}(f) R_{n,m}(r) \end{aligned} \tag{2.3}$$

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

$$Z_{n,m}^{L}(f) = -(Z_{n,m}^{R}(f))^{*}$$

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

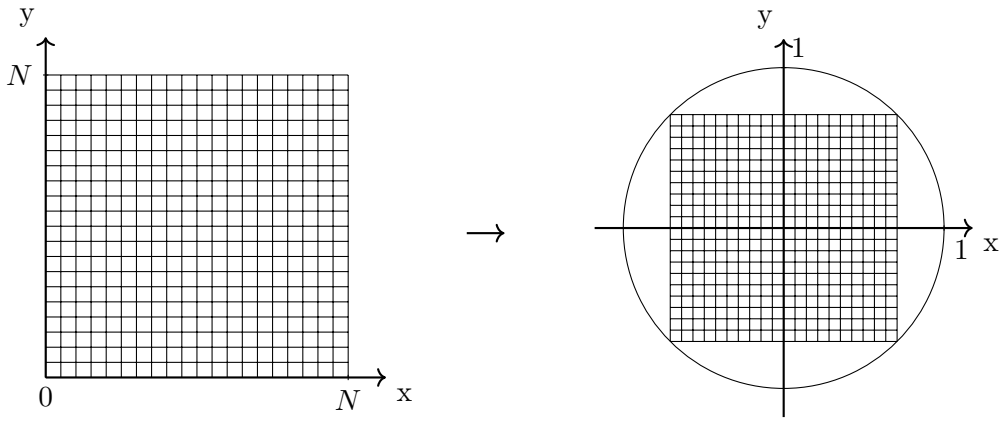
The conventional Zernike moments and the QZMs are defined in terms of polar coordinates, for image functions, whose domain is the unit disk. On the other hand, digital images are usually defined in image coordinates, with the coordinates being integers ranging from 0 to $N-1$ (the number of pixels along 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 [15].

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 calcu-



(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

lated by the following formulas:

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

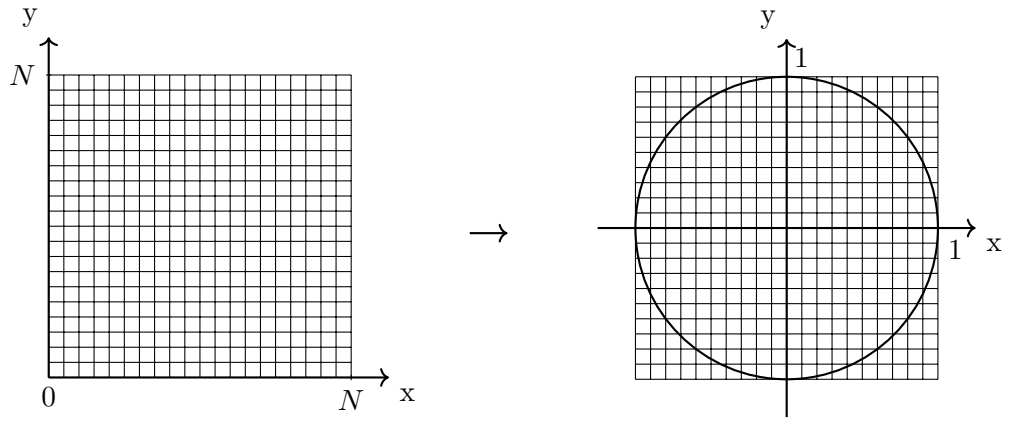
where c_1, $c_2 \in \mathbb{R}$ depend on which of the previously mentioned transformations is used. A $\lambda \in \mathbb{R}$ scaling factor is also defined for each transformation. For the first one (Figure 2.1): $c_1 = \frac{\sqrt{2}}{N-1}$, $c_2 = -\frac{1}{\sqrt{2}}$, and $\lambda = \frac{2}{\pi}$.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

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.



(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

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.

2.5 Quaternion Zernike Moment Invariants (QZMIs)

Chen et al. [14, 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 [14], 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 $\Psi^m_{n,k} = L^R_{n,m}(f)(L^R_{k,m}(f))^*$, 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, $\overline{\Psi}^m_{n,k} = \overline{L}^R_{n,m}(f)(\overline{L}^R_{k,m}(f))^*$ 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 [16] and relying on the Numpy [17] and Numba [18] libraries to achieve efficient and fast computation of the moment invariants, as well as the Python Imaging Library (PIL, Pillow) [19] 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

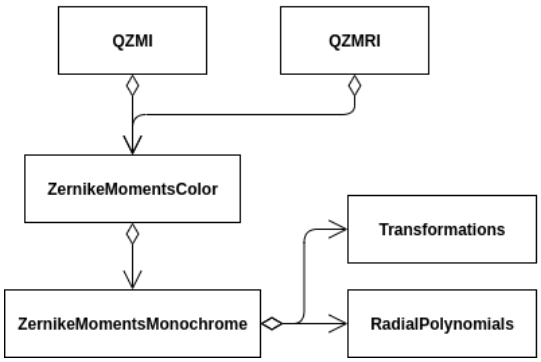


Figure 4.1: The relationships between the classes used to calculate QZMIs and QZMRIs

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 [15]. 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 basis 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. [20]. 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.

Computing the discrete measure

To calculate the Zernike moments over the discrete orthogonal points system, instead of the λ scaling factor used in the linear transformation, a different measure is used. To calculate this $\mu_{k,N}$ measure (defined in), the following integral has to be calculated:

$$\mathcal{A}_{k,N} = \int_{-1}^{1} \ell_{k,N}(x)\,\mathrm{d}x$$

where $\ell_{k,N}$ is the k^{th} Lagrange basis polynomial corresponding to the roots of the Legendre polynomial P_N.

The Gauss-Legendre quadrature is based on the roots of the Legendre polynomial P_n of degree n. Let $x_1, x_2, \ldots, x_n$ be the roots of P_n.

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx \sum_{k=1}^{n} w_k f(x_k)$$

$$w_k = \frac{2}{(1-x_k^2)(P_n'(x_k))^2}$$

The quadrature is exact for all polynomials whose degree is at most $2n-1$. Now $\ell_{k,N}(x)$ is an $N-1$ degree polynomial, so the Gauss-Legendre quadrature with $n = N$ is exact

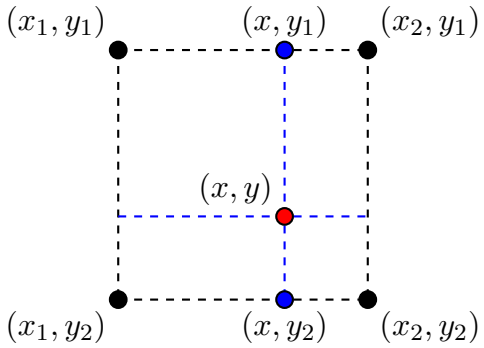


Figure 4.3: The point (x, y) and its neighboring points, which are used to approximate $f(x, y)$ by bilinear interpolation

for $\ell_{k,N}$. Furthermore, $\ell_{k,N}$ is defined such that $\ell_{k,N}(x_i) = 0$ for $i \neq k$, and $\ell_{k,N}(x_k) = 1$ Thus:

$$\int_{-1}^{1} \ell_{k,N}(x)\,\mathrm{d}x = w_k = \frac{2}{(1 - x_k^2)(P_n'(x_k))^2}$$

Since the roots of the Legendre polynomial P_N must be computed to obtain the points system, this formula gives an easy and fast way to calculate the exact values of $\mu_{k,N}$.

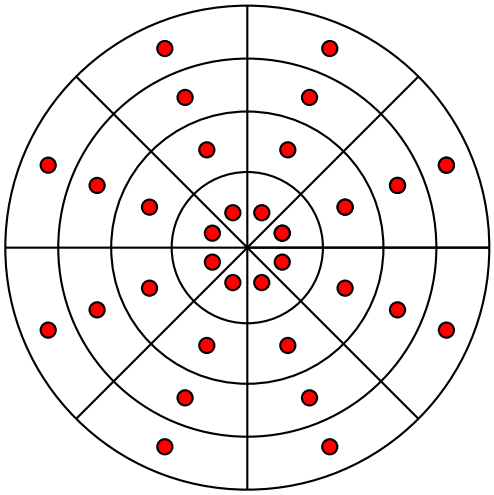
Image interpolation

After obtaining the discrete orthogonal points system the values of the image function have to be approximated at each point, because the original pixel values do not fall exactly on the new points.

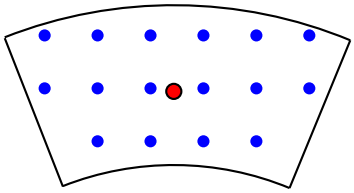
First, the image is linearly transformed onto the unit disk using the transformation shown on Figure 2.2, so that the the transformed image covers the entire unit disk. This transformation is used as opposed to the one on Figure 2.1, because the points system covers the entire unit disk, so the image also has to cover the whole disk.

There are two ways to approximate the values at each point, depending on the number of points in the discrete orthogonal points system.

Approximately the same number of points as pixels. If the number of points is approximately the same as the number of pixels in the image, then bilinear interpolation can be used for each point.



(a) The points system on the unit disk and the annular sectors over which the pixel values are averaged.



(b) A single annular sector with the original image pixels (blue) and the point in which the approximate function value has to be calculated (red).

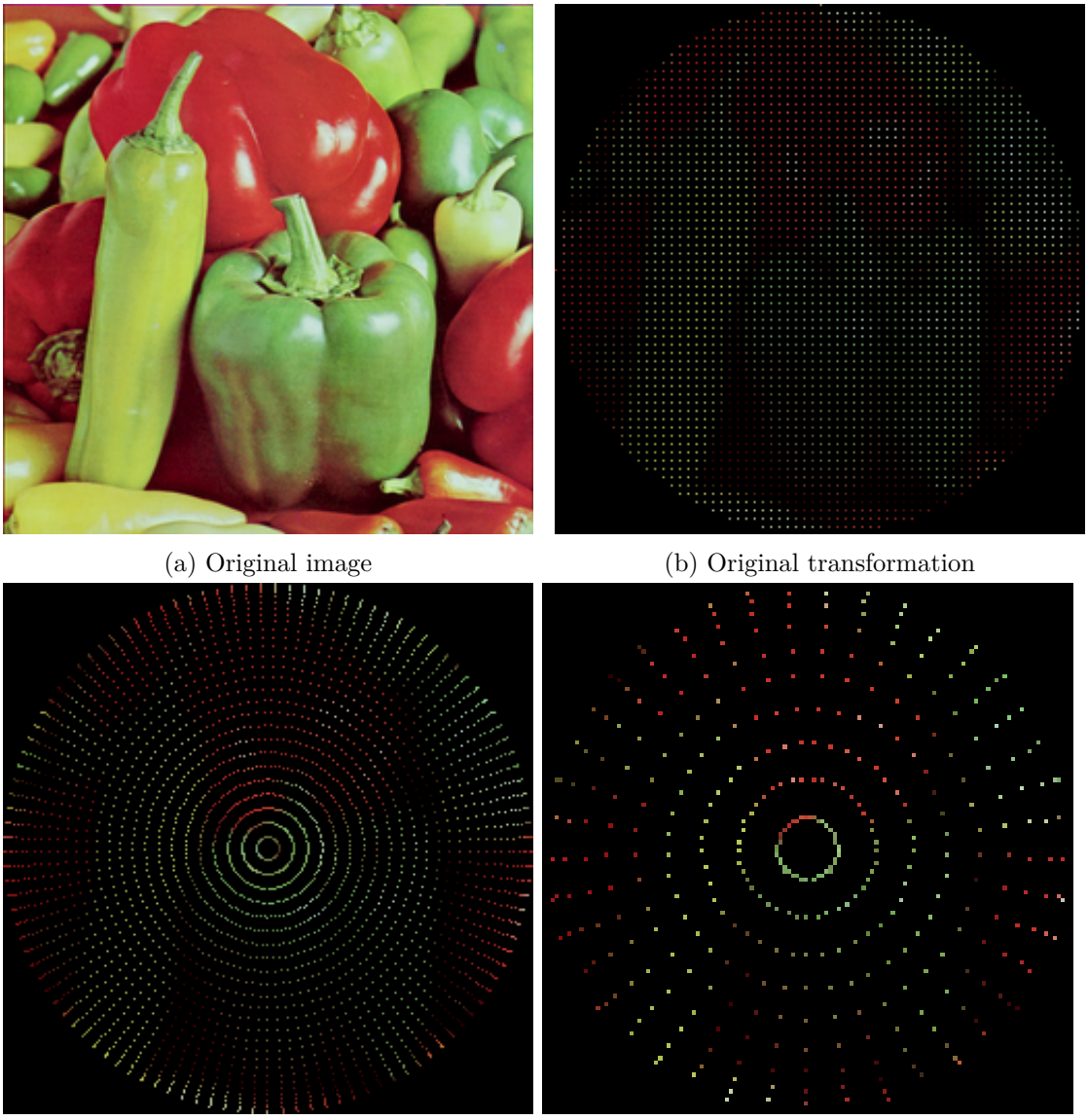
Figure 4.4

First, the four pixels of the image that are closest to the given point (x, y) are determined. These four points form a square. Then, the weighted average of these points is calculated along the x axis, giving approximate function values at two points, which only differ in their y coordinate. Finally, the weighted average of the function values at these points is calculated to get the approximation of $f(x, y)$. The points involved in this interpolation are shown on Figure 4.3.

Much less points than the number of pixels. If the number of points is far fewer than the number of pixels, then using bilinear interpolation many of the original pixel values would not be represented in the final approximate function values. Therefore, the following algorithm is used to approximate function values using discrete integration.

First, for each pixel in the original image, determine which point it falls closest to after the linear transformation. Then, for each point in the discrete orthogonal points system, calculate the average of the pixels that are closest to that point. This algorithm divides the unit disk into annular sections based on the proximity to the new points. This is shown on Figure 4.4.

The comparison between one of the original transformations and the two methods for approximating function values is shown on Figure 4.5 on the peppers image from the USC-SIPI Image Database [21].



(a) Original image

(b) Original transformation

(c) Bilinear interpolation

(d) Interpolation by integrals

Figure 4.5: Different transformations with different interpolation methods of the peppers image.

Chapter 5

Tests and comparison with the previous method

In this chapter we present the different tests that were performed to compare the capabilities of the old and new methods. For each test, the results of both methods of discretization are compared.

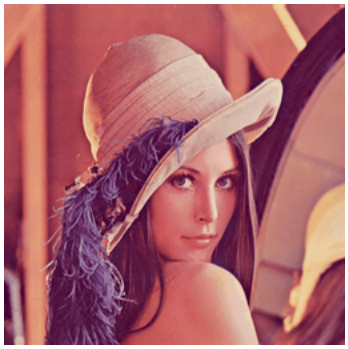
In total, four kinds of tests were conducted:

- Invariance
- Image reconstruction
- Image recognition
- Template matching

5.1 Test images

The images for testing were acquired from multiple online image libraries. The Lenna and Pepper images [21] (shown on Figure 5.1) were used to test image reconstruction as well as to demonstrate the different points systems.

For the image recognition tests, two sets of images were used. The first set consists of 14 images chosen from the Columbia Object Image Library (COIL-100) [22], shown on Figure 5.2. These images are originally 128×128 pixels, but they were placed on a



(a)

(b)

Figure 5.1: The Lenna and Pepper images

204×204 black background so that the rotated, scaled and translated versions of the images remain completely within these dimensions.

A set of 1008 rotated images was created by rotating each of the 14 images by a degree $\alpha \in \{0, 5, 10, \ldots, 350, 355\}$. Some examples of the extended and rotated images are shown on Figure 5.3.

Another set of 1176 rotated, scaled and translated images was created by translating each image by -11 pixels in the x direction and 9 pixels in the y direction. Then the translated images were rotated by $\alpha \in \{0, 30, 60, \ldots, 300, 330\}$. Finally, each rotated and translated image was scaled by $\lambda \in \{0.5, 0.75, \ldots, 1.75, 2\}$. When either scaling or rotation required, bilinear interpolation was used. Some examples of the RST transformed images are shown on Figure 5.4.

Another set of 13 images was acquired from the Amsterdam Library of Object Images (ALOI) [23]. These are shown on Figure 5.5. Originally, these size of these images was 768×576 pixels, but the were downscaled to 96×72 and subsequently extended to 152×128 by placing the images on a black background. Similarly to the test sets created using the COIL-100 images, the ALOI images were also translated, rotated and scaled, yielding a set of 1092 RST transformed images. The parameters of the transformation were the same as for the COIL-100 images, except for the translation, where $\Delta x = 8$ and $\Delta y = 5$ was used.

(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

(k) (l) (m) (n)

Figure 5.2: The 14 selected images from the Columbia Object Image Library (COIL-100)

(a) $\alpha = 0°$ (b) $\alpha = 35°$ (c) $\alpha = 255°$

Figure 5.3: Some extended and rotated images from COIL

(a) $\alpha = 0°$, $\lambda = 1$

(b) $\alpha = 150°$, $\lambda = 0.5$

(c) $\alpha = 270°$, $\lambda = 1.5$

Figure 5.4: Some RST transformed images from COIL. All images are translated by $\Delta x = -11$, $\Delta y = 9$

(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

(k) (l) (m)

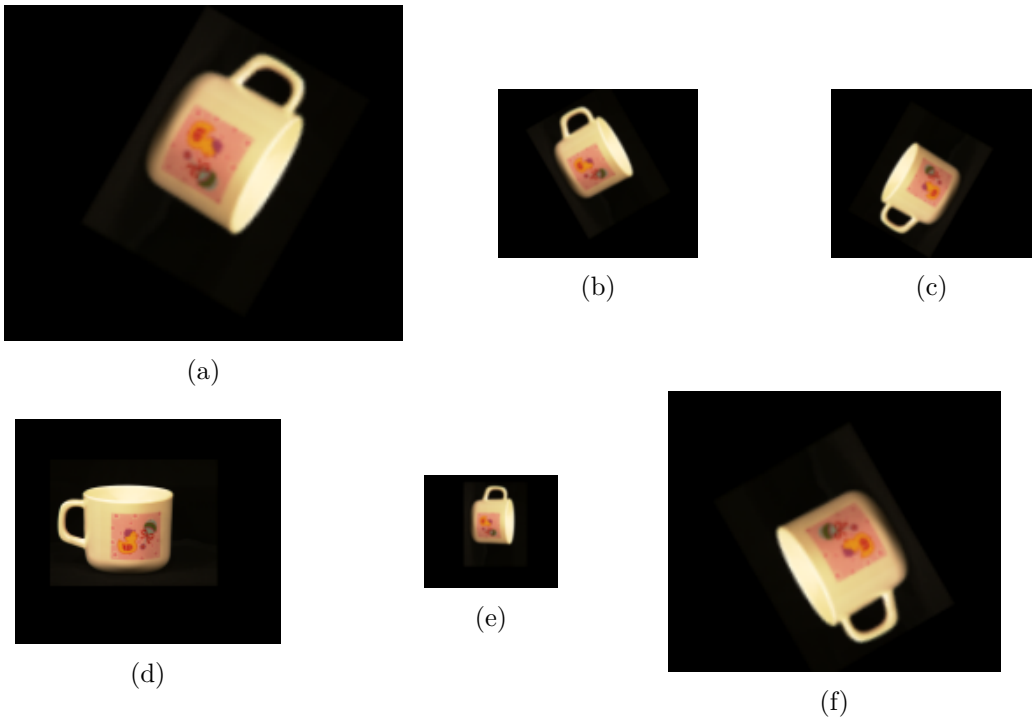
Figure 5.5: The 13 selected images from the Amsterdam Library of Object Images (ALOI)

5.2 Invariance test

In order to test the invariance of the quaternion Zernike moment invariants with respect to rotation, scaling and translation, the QZMIs of order 1 to 4 were calculated for a given image and all of its RST transformations. Then, the modulus of these QZMIs was calculated, as well as the mean (μ), standard deviation (σ) and $\frac{\sigma}{\mu}$ for the same moment of all transformed images.

Results

The modulus of the QZMIs for the transformed images shown on Figure 5.6 is shown in Table 5.1 for the old method of discretization and in Table 5.2 for the new, proposed method of discretization. The coefficient of variation ($\frac{\sigma}{\mu}$) shows that using both methods, the moments are invariant to RST transformation. The only rows where $\frac{\sigma}{\mu}$ is higher are the ones where the modulus of the moment is very close to zero, and thus small numerical errors impact this number significantly. Comparing the two methods, the proposed one yields slightly lower values for the coefficient of variation for all moments.



(a) (b) (c) (d) (e) (f)

Figure 5.6: The images with different rotation and scale, which are used to test the invariance of the methods.

	Fig.5.6a	Fig.5.6b	Fig.5.6c	Fig.5.6d	Fig.5.6e	Fig.5.6f	$\frac{\sigma}{\mu}$
$\lvert\overline{\Psi}^1_{1,1}\rvert$	1.50e-2	1.58e-2	1.54e-2	1.61e-2	1.58e-2	1.49e-2	3.73%
$\lvert\overline{\Psi}^0_{2,0}\rvert$	2.965	2.964	2.964	2.963	2.964	2.965	0.028%
$\lvert\overline{\Psi}^0_{2,2}\rvert$	8.793	8.785	8.789	8.783	8.786	8.794	0.057%
$\lvert\overline{\Psi}^2_{2,2}\rvert$	1.50e-4	1.65e-4	1.57e-4	1.69e-4	1.65e-4	1.47e-4	6.87%
$\lvert\overline{\Psi}^1_{3,1}\rvert$	5.91e-2	6.25e-2	6.09e-2	6.34e-2	6.22e-2	5.89e-2	3.71%
$\lvert\overline{\Psi}^1_{3,3}\rvert$	0.233	0.247	0.241	0.250	0.246	0.233	3.69%
$\lvert\overline{\Psi}^3_{3,3}\rvert$	1.43e-6	1.61e-6	1.50e-6	1.67e-6	1.62e-6	1.38e-6	9.40%
$\lvert\overline{\Psi}^0_{4,0}\rvert$	4.827	4.821	4.824	4.820	4.822	4.828	0.086%
$\lvert\overline{\Psi}^0_{4,2}\rvert$	14.316	14.291	14.302	14.284	14.292	14.318	0.114%
$\lvert\overline{\Psi}^2_{4,2}\rvert$	7.39e-4	8.13e-4	7.74e-4	8.35e-4	8.12e-4	7.26e-4	6.85%
$\lvert\overline{\Psi}^0_{4,4}\rvert$	23.309	23.249	23.276	23.232	23.251	23.314	0.171%
$\lvert\overline{\Psi}^2_{4,4}\rvert$	3.64e-3	4.00e-3	3.81e-3	4.11e-3	4.00e-3	3.58e-3	6.83%
$\lvert\overline{\Psi}^4_{4,4}\rvert$	1.28e-8	1.52e-8	1.39e-8	1.59e-8	1.51e-8	1.25e-8	12.00%

Table 5.1: The modulus of QZMIs using the old method for discretization. Note that $\frac{\sigma}{\mu}$ was calculated using the QZMIs for all transformation of the image, not juts the values shown in the table.

	Fig.5.6a	Fig.5.6b	Fig.5.6c	Fig.5.6d	Fig.5.6e	Fig.5.6f	$\frac{\sigma}{\mu}$
$\lvert\overline{\Psi}^1_{1,1}\rvert$	1.49e-2	1.58e-2	1.54e-2	1.60e-2	1.57e-2	1.49e-2	3.72%
$\lvert\overline{\Psi}^0_{2,0}\rvert$	2.965	2.964	2.964	2.963	2.964	2.965	0.028%
$\lvert\overline{\Psi}^0_{2,2}\rvert$	8.793	8.785	8.789	8.783	8.786	8.793	0.056%
$\lvert\overline{\Psi}^2_{2,2}\rvert$	1.49e-4	1.64e-4	1.55e-4	1.68e-4	1.63e-4	1.47e-4	6.82%
$\lvert\overline{\Psi}^1_{3,1}\rvert$	5.90e-2	6.25e-2	6.08e-2	6.33e-2	6.21e-2	5.89e-2	3.70%
$\lvert\overline{\Psi}^1_{3,3}\rvert$	0.233	0.247	0.240	0.250	0.245	0.233	3.68%
$\lvert\overline{\Psi}^3_{3,3}\rvert$	1.41e-6	1.61e-6	1.48e-6	1.65e-6	1.60e-6	1.37e-6	9.32%
$\lvert\overline{\Psi}^0_{4,0}\rvert$	4.828	4.821	4.824	4.820	4.822	4.828	0.085%
$\lvert\overline{\Psi}^0_{4,2}\rvert$	14.317	14.291	14.304	14.286	14.293	14.318	0.113%
$\lvert\overline{\Psi}^2_{4,2}\rvert$	7.35e-4	8.13e-4	7.68e-4	8.29e-4	8.07e-4	7.25e-4	6.80%
$\lvert\overline{\Psi}^0_{4,4}\rvert$	23.312	23.249	23.279	23.236	23.254	23.314	0.170%
$\lvert\overline{\Psi}^2_{4,4}\rvert$	3.62e-3	4.00e-3	3.79e-3	4.09e-3	3.98e-3	3.58e-3	6.78%
$\lvert\overline{\Psi}^4_{4,4}\rvert$	1.28e-8	1.52e-8	1.37e-8	1.56e-8	1.50e-8	1.24e-8	11.94%

Table 5.2: The modulus of QZMIs using the new method for discretization. Note that $\frac{\sigma}{\mu}$ was calculated using the QZMIs for all transformation of the image, not juts the values shown in the table.

5.3 Image reconstruction

As described in Section 2.3, the quaternion Zernike moments of an image can be used to approximately reconstruct the original image using the formulas in (2.3). This reconstruction requires QZMs of up to a finite degree M.

In order to calculate the reconstructed discrete image, the

$$f(r_{x,y}, \theta_{x,y}) \approx \sum_{n=0}^{M} \sum_{m=-n}^{n} Z_{n,m}^{R}(f) R_{n,m}(r_{x,y}) e^{\boldsymbol{\mu} m \theta_{x,y}}$$

formula was used for each pixel with image coordinates (x, y), where $(r_{x,y}, \theta_{x,y})$ are the polar coordinates obtained by performing the linear transformation of the image onto the unit disk, using the transformation shown on Figure 2.1 (tf_1) for the old method and the transformation shown on Figure 2.2 (tf_2) for the new one. The reason for this difference in transformation is that for the proposed discrete orthogonal points system, the interpolated pixel values are calculated using tf_2, while conventionally for (quaternion) Zernike moments, tf_1 is used [2].

To measure the error of the reconstruction the normalized mean squared error (ε^2) was used. If $f(x, y)$ is the original and $\widehat{f}(x, y)$ is the reconstructed image, both with size $N \times N$, then the normalized mean squared error is defined as:

$$\varepsilon^2 = \frac{\sum_{x=1}^{N} \sum_{y=1}^{N} \left| f(x, y) - \widehat{f}(x, y) \right|^2}{\sum_{x=1}^{N} \sum_{y=1}^{N} \left| f(x, y) \right|^2}$$

In the cases, where tf_2 is used only the part of the image falling inside the unit circle is reconstructed and thus the mean squared error is calculated over only this part of the image.

For the new method of discretization the number of points on the unit disk was chosen to be approximately the same as the number of pixels falling inside the inscribed circle of the image.

Results

Image reconstruction was performed for both the Lenna and the Pepper images [21], using image sizes ranging from 64×64 up to 256×256. Depending on the size of the

M	50	100	150	250	350
old method ε^2	0.02659	0.01341	0.00868	0.00428	0.00279
new method ε^2	0.01611	0.00790	0.00463	0.00190	0.00238
change	-39.4%	-41.1%	-46.7%	-55.6%	-14.7%

Table 5.3: Comparison of the normalized mean squared errors between the two method, for the 256×256 Lenna image.

image, QZMs of up to degree 350 were used reconstruct the image. However, in the case of smaller images, for example a 64×64 image, only QZMs of up to 100 degree could be used, as using higher degrees would result in trying to extract more information from the original image than there is information contained in the pixel values, thus increasing the error of reconstruction. This phenomenon is visible, for example on Figure 5.8 for the 64×64 Lenna, M = 100 case.

Figure 5.7 shows some of the reconstructed images and their normalized mean squared error when using the old method of discretization, while Figure 5.8 shows the same data using the new method.

Comparing the error of reconstruction between the two methods, using the discrete orthogonal points system provides much lower normalized mean squared errors for almost all levels of reconstruction. Table 5.3 shows the comparison between the ε^2 values and the change in the error when using the proposed method instead of the old one. The decrease in the error of reconstruction is is significant for both low and high maximal degrees of QZMs.

Due to the lower number of points used by the second method (because of the different transformation of the image onto the unit disk) the previously described increase in reconstruction error occurs for lower Ms than in the original method.

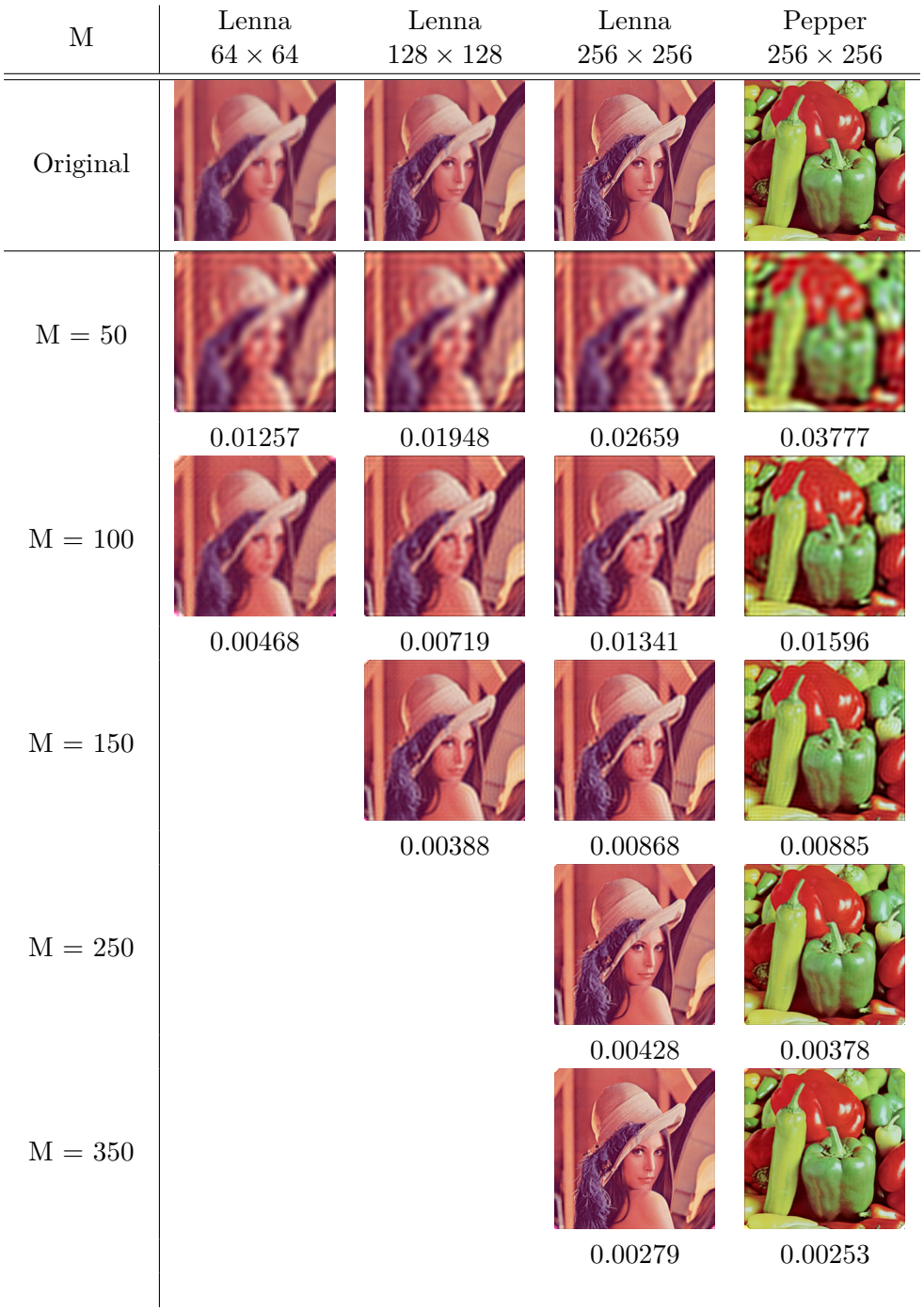


Figure 5.7: Reconstructed images using the old method, with the normalized mean squared error shown below each image.

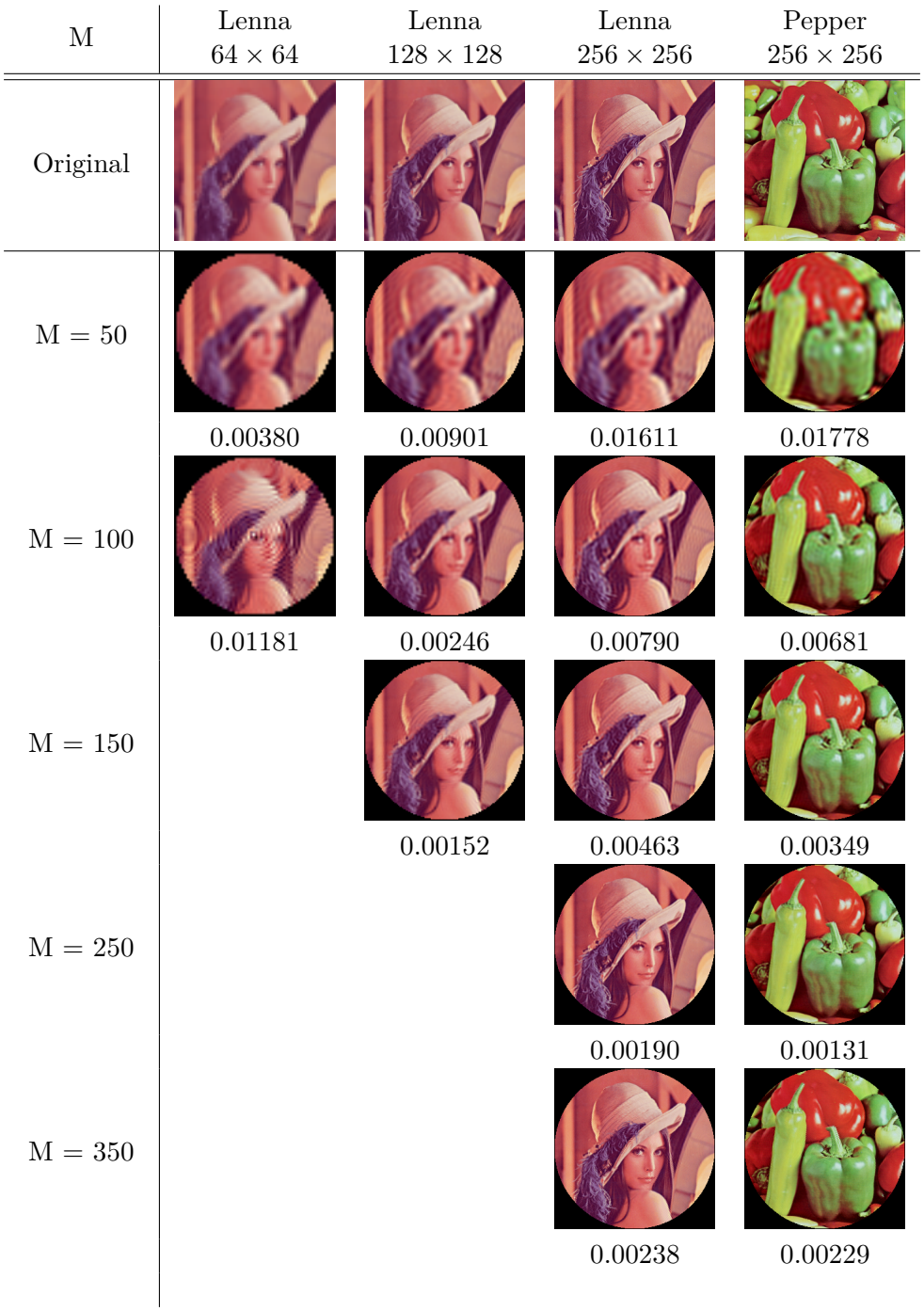


Figure 5.8: Reconstructed images using the proposed new method, with the normalized mean squared error shown below each image.

5.4 Image recognition

In order to test the image recognition capabilities of the methods, the COIL-100 [22] and ALOI [23] images and the RST transformed sets of images were used. The exact parameters for the transformations performed to obtain these sets of images is described in Section 5.1. The goal of the image recognition test is to see what percentage of the transformed images can each method correctly identify from the original, non-transformed images.

To recognize an image, first the QZMIs have to be calculated. In this thesis, we used QZMIs of up to degree 4, but not all possible QZMIs were used. The selected QZMIs were: $\overline{\Psi}^1_{1,1}, \overline{\Psi}^0_{2,0}, \overline{\Psi}^2_{2,2}, \overline{\Psi}^1_{3,1}, \overline{\Psi}^3_{3,3}, \overline{\Psi}^0_{4,0}, \overline{\Psi}^2_{4,2}, \overline{\Psi}^4_{4,4}$. These 8 quaternion valued invariants contain a total of 20 real valued invariants, since while each quaternion could provide of 4 real invariants, the QZMIs $\overline{\Psi}^m_{n,k}$ with $n = k$ necessarily have $\operatorname{Im} \overline{\Psi}^m_{n,n} = 0$. This follows directly from the definition (see Section 2.5):

$$\overline{\Psi}^m_{n,n} = \overline{L}^R_{n,m}(f)(\overline{L}^R_{n,m}(f))^* = \left|\overline{L}^R_{n,m}(f)\right|^2 \in \mathbb{R}$$

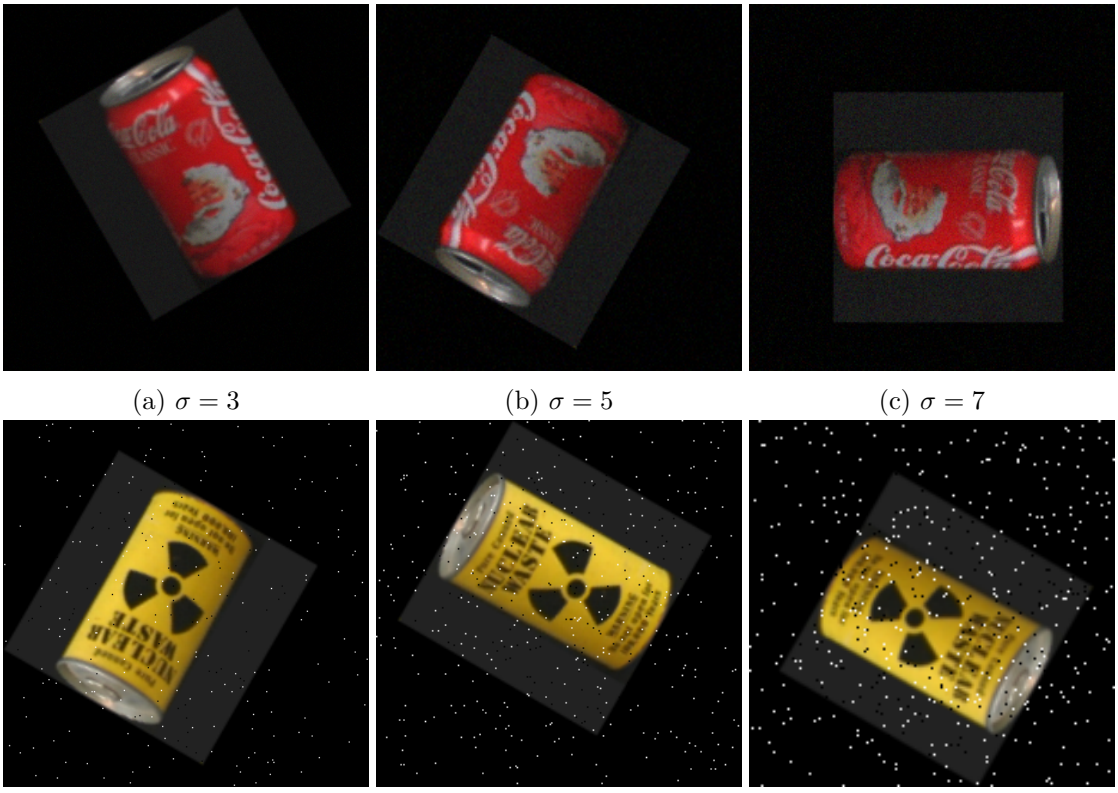
Thus from the selected 8 QZMIs four provide a single real-valued invariant and another four provide 4 real-valued invariants. These 20 real invariants are then used to construct a vector I of length 20. This vector is normalized using the method presented by Suk and Flusser [11].

$$I_k = sgn(I_k)\,|I_k|^{\frac{1}{2}} \qquad (k = 1, 2, \ldots, 20)$$

This invariant vector is calculated for an RST transformed image and all of the original images. Then the minimum Euclidean distance is used to choose from the original images. The recognition algorithm classifies the transformed image as this chosen image.

Noise generation

To test the robustness of the methods against noise the tests were also performed with different levels of real-valued Gaussian noise added to all images. The parameters of the Gaussian noise were as follows: the mean of the distribution was always at 0, while standard deviation (σ) values from 1 up to 60 were used for the QZMIs and standard deviation ranging from 40 to 120 was used to test the QZMRIs. QZMRIs are more robust against noise than QZMIs, so higher noise values could be used. Instead of using Gaussian noise, for some test cases salt-and-pepper noise was added with densities (p) ranging from 0.2% up to 15% for the QZMIs and ranging from 5% to 30% for the QZMRIs. Some examples for both types of noise are shown on Figure 5.9.



(a) $\sigma = 3$ (b) $\sigma = 5$ (c) $\sigma = 7$

(d) $p = 0.4\%$ (e) $p = 1\%$ (f) $p = 3\%$

Figure 5.9: Some test images with Gaussian noise (top row) or salt-and-pepper noise (bottom row) with different parameters added to them.

The tests were performed directly using the raw, noisy images, without any kind of filtering applied to the image. The goal was to see the robustness of these methods against noise.

Test cases

The recognition tests were performed using both the old and the proposed method for discretization. Furthermore, the new method was tested with with two different N values for creating the discrete orthogonal system of $N(4N + 1)$ points.

In the first case N was chosen such that the total number of points in the system should

be approximately the same as the pixels of the image falling inside its inscribed circle.

$$\begin{aligned} N &\in \mathbb{N} \\ (4N+1)N &\approx n^2\frac{\pi}{4} \\ N &= \left\lfloor \frac{\sqrt{4n^2\pi+1}-1}{8} \right\rfloor \end{aligned}$$

where the size of the image is $n \times n$. Based on the interpolating methods described in Section 4.3 this means that bilinear interpolation should be used.

In the other case N was chosen such that the number of points is the minimum required for the method to be accurate. In the case of using QZMIs up to degree 4, this means $N = 10$. In this case, based on Section 4.3 the interpolation should be done using the integral interpolation formula.

Note that for the old method, the number of points used is the same as the number of pixels in the image.

Three sets of transformed images were tested, which were generated as described in Section 5.1:

- COIL-100 rotated (1008 images)
- COIL-100 rotated, scaled, translated (1176 images)
- ALOI rotated, scaled, translated (1092 images)

For the set where only rotation was applied to the images, instead of using QZMIs the rotation invariant QZMRIs were used.

Results

When the images have Gaussian noise, the new method with the higher number of points performs the best for all levels of noise, far outperforming the original method in recognition capabilities. The rate of recognition for Gaussian noise in case of the RST transformed image sets is shown in Table 5.4. The recognition rate of the new method remains relatively high (around 80%) even for high noise values, while the recognition rate of the original method drops significantly (below 20%).

When using only the minimal possible number of points in the new method, for all noise levels the recognition rate remains roughly around the same level as with the original

Image set	**Noise stdev.**	**Old method** (%)	**New method** many points (%)	**New method** few points (%)
COIL - RST	No noise	99.06	99.15	
	1	98.98	99.49	
	2	98.98	99.74	
	3	98.55	99.83	
	5	95.15	99.49	
	7	95.15	98.72	
	9	76.87	98.47	
	40	52.89	88.52	
	50	48.21	84.10	
	60	41.58	85.80	
ALOI - RST	No noise	99.91	100.00	94.60
	1	94.51	99.08	86.63
	2	84.89	93.13	86.35
	3	78.85	88.55	74.81
	5	72.07	93.31	58.24
	7	63.28	94.23	61.81
	9	55.04	94.32	48.81
	40	18.41	90.84	15.29
	50	19.32	82.51	
	60	13.19	84.89	

Table 5.4: Percentage of images recognized correctly by QZMIs in the case of using the RST transformed image sets with added Gaussian noise.

Noise type	**Noise param.**	**Old method** (%)	**New method** many points (%)	**New method** few points (%)
No noise	100.00	100.00		
Gauss	40	91.37	99.90	
	50	86.51	100.00	
	60	84.42	99.60	
	70	80.85	99.90	
	80	76.98	98.12	
	90	75.79	99.01	
	100	64.98	98.91	
	110	64.78	97.62	
	120	68.55	96.23	
Salt	5	100.00	100.00	
	10	100.00	100.00	
	15	100.00	100.00	
	20	100.00	100.00	
	25	100.00	100.00	
	30	100.00	100.00	

Table 5.5: Percentage of images recognized correctly by QZMRIs in the case of using the rotated COIL image set with either Gaussian or salt-and-pepper noise. The noise parameter means the standard deviation in case of the Gaussian noise, and the density in case of the salt-and-pepper noise.

method, but the computation required to achieve such levels of image recognition is much lower because of the lower number of points.

The same observation can be made about the QZMRIs used to recognize the rotated images. Even for extremely high noise values, the recognition rate of the new method remains well above 90%. The results for the QZMRIs are shown in Table 5.5.

When salt-and-pepper noise is added to the images, the recognition capabilities of both methods are high, but no clear difference is visible between them. Even using the minimal possible number of points in the new method, the recognition rates are almost the same as with the other two methods. These same observation can be made about the recognition capabilities of the QZMRIs. The results for salt-and-pepper noise are shown for QZMIs and QZMRIs in Table 5.6 and Table 5.5 respectively.

Image set	**Noise density**	**Old method** (%)	**New method** many points (%)	**New method** few points (%)
COIL - RST	No noise	99.06	99.15	
	0.2	99.66	99.32	
	0.4	99.91	99.74	
	0.6	99.91	99.91	
	1	98.98	99.91	
	2	99.66	93.96	
	3	99.40	99.40	
	5	97.87	94.90	
	10	99.91	93.03	
	15	99.91	93.20	
ALOI - RST	No noise	99.91	100.00	94.60
	0.2	88.64	90.75	
	0.4	86.08	91.30	
	0.6	83.97	90.11	
	1	95.97	94.60	
	2	98.44	94.96	
	3	97.61	96.06	
	5	97.99	97.25	
	10	98.44	87.27	
	15	93.50	91.30	

Table 5.6: Percentage of images recognized correctly by QZMIs in the case of using the RST transformed image sets with added salt-and-pepper noise.

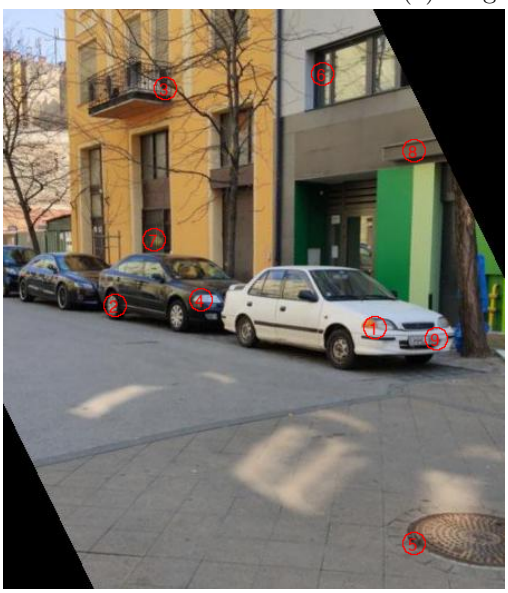
5.5 Template matching

This final test aims to present some possible application of the pattern matching capabilities of the QZMIs. Some pictures were taken using a the camera on a Xiaomi Mi 9 smartphone with autofocus enabled. The images were taken of the same scene but with different focus and different rotation of the camera.

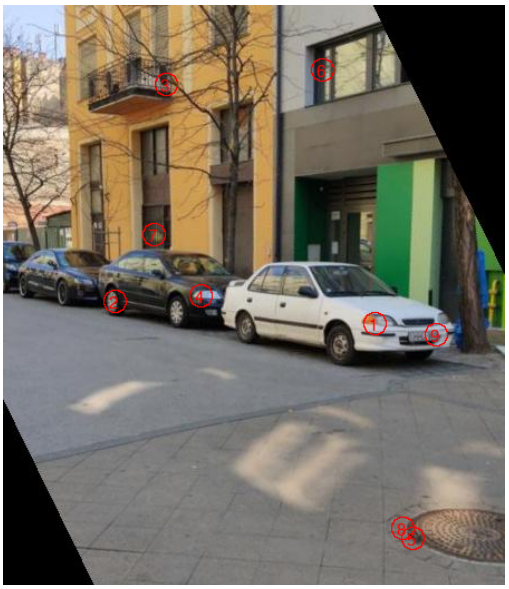
A total of 9 circles with radii of 10 pixels were chosen so that they represent some unique area on the pictures. Then, on another picture taken with different rotation and focus, after determining the scaling factor between the two images, a sliding circular window was moved across the second image. The invariant vector described in the image recognition test was constructed for each window, and then the minimal Euclidean distance was used to find each area of the original image on the second image.

The original image and the results of both methods are shown on Figure 5.10. Both methods managed to correctly find 8 out of the 9 templates on the other image. Neither managed to correctly locate template number 8.

(a) Original image



(b) Old method



(c) New method

Figure 5.10: The original image with the templates and the result of both methods.

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] William Rowan Hamilton. *Elements of Quaternions*. Longmans Green, London, U.K., 1866.

[13] T. A. Ell and S. J. Sangwine. Hypercomplex fourier transforms of color images. *IEEE Transactions on Image Processing*, 16(1):22–35, 2007.

[14] 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.

[15] 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.

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

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

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

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

[20] 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.

[21] The USC-SIPI Image Database. http://sipi.usc.edu/database. [Accessed: 2020.05.09.].

[22] S. A. Nene, S. K. Nayar, and H. Murase. Columbia Object Image Library (COIL-100). *Technical Report CUCS-006-96*, 1996.

[23] J.M. Geusebroek, G.J. Burghouts, and A.W.M. Smeulders. ALOI: Amsterdam Library of Object Images. *International Journal of Computer Vision*, 61(1):103–122, 2005. URL `http://aloi.science.uva.nl/`.