HAL
open science

Fast generic polar harmonic transforms

Thai V. Hoang, Salvatore Tabbone

To cite this version:

Thai V. Hoang, Salvatore Tabbone. Fast generic polar harmonic transforms. IEEE Transactions on Image Processing, 2014, 23 (7), pp.2961 - 2971. 10.1109/TIP.2014.2322933 . hal-01083716

HAL Id: hal-01083716

https://inria.hal.science/hal-01083716v1

Submitted on 17 Nov 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Fast Generic Polar Harmonic Transforms

Thai V. Hoang and Salvatore Tabbone

***Abstract*—Generic polar harmonic transforms have recently been proposed to extract rotation-invariant features from images and their usefulness has been demonstrated in a number of pattern recognition problems. However, direct computation of these transforms from their definition is inefficient and is usually slower than some efficient computation strategies that have been proposed for other competing methods. This paper presents a number of novel strategies to compute these transforms rapidly. The proposed methods are based on the inherent recurrence relations among complex exponential and trigonometric functions used in the definition of the radial and angular kernels of these transforms. The employment of these relations leads to recursive and addition chain-based strategies for fast computation of harmonic function-based kernels. Experimental results show that the proposed method is about $10\times$ faster than direct computation and $5\times$ faster than fast computation of Zernike moments using the q-recursive strategy. Thus, among all existing rotation-invariant feature extraction methods, polar harmonic transforms are the fastest.**

***Index Terms*—Polar harmonic transforms; orthogonal moments; recurrence relation; shortest addition chain; Chebyshev polynomials; fast computation**

I. Introduction

Image moments extracted from a unit disk region are usually used in invariant pattern recognition problems as rotation-invariant features [1]. Let f be the image function, its moment H_{nm} of order n and repetition m $(n, m \in \mathbb{Z})$ over the unit disk region is computed as

$$H_{nm} = \iint\limits_{x^2+y^2\leq 1} f(x,y)V^*_{nm}(x,y)\,\mathrm{d}x\mathrm{d}y, \tag{1}$$

where V_{nm} is the decomposing kernel and the asterisk denotes the complex conjugate. It was shown in [2] that in order to have the "invariance in form" property on H_{nm}, the polar form of the kernel, $V_{nm}(r,\theta)$, should take the form $R_n(r)A_m(\theta)$ with $A_m(\theta) = \mathrm{e}^{im\theta}$. In addition, orthogonality is often preferred in many practical situations to reduce information redundancy in compression, to avoid difficulties in image reconstruction, and to increase accuracy in pattern recognition. Orthogonality between kernels means $\langle V_{nm}, V_{n'm'}\rangle = \int_0^1 R_n(r)R^*_{n'}(r)r\,\mathrm{d}r \int_0^{2\pi} A_m(\theta)A^*_{m'}(\theta)\,\mathrm{d}\theta = \delta_{nn'}\delta_{mm'}$. From the orthogonality between angular kernels:

$$\int_0^{2\pi} A_m(\theta)A^*_{m'}(\theta)\,\mathrm{d}\theta = 2\pi\delta_{mm'}, \tag{2}$$

the remaining condition for radial kernels is

$$\int_0^1 R_n(r)R^*_{n'}(r)r\,\mathrm{d}r = \frac{1}{2\pi}\,\delta_{nn'}. \tag{3}$$

T. V. Hoang is with Inria Nancy - Grand Est, 615 rue du Jardin Botanique, 54600 Villers-lès-Nancy, France

S. Tabbone is with Université de Lorraine, LORIA, Campus scientifique, 54506 Vandœuvre-lès-Nancy, France

The above equation represents the condition for the definition of a set of radial kernels in order to have orthogonality between kernels. There exists a number of methods that satisfies this condition. One direction is to employ polynomials of the variable r for R_n, which turn out to be special cases of the Jacobi polynomials [3], [4]. Popular methods are Zernike moments (ZM) [5], pseudo-Zernike moments (PZM) [6], orthogonal Fourier–Mellin moments (OFMM) [7], Chebyshev–Fourier moments [8], and pseudo Jacobi–Fourier moments (PJFM) [9] (see [10, Section 6.3] or [11, Section 3.1] for a comprehensive survey). Despite its popularity, this class of orthogonal moments involves factorial computation, which results in high computational complexity and numerical instability, which often limit its practical usefulness.

The second direction defines V_{nm} as the eigenfunctions of the Laplacian ∇^2 on the unit disk, similar to the interpretation of Fourier basis as the set of eigenfunctions of the Laplacian on a rectangular domain. These eigenfunctions are obtained by solving the Helmholtz equation, $\nabla^2 V + \lambda^2 V = 0$, in polar coordinates to have the radial kernels defined based on Bessel functions of the first and second kinds [12]. By imposing the condition in Equation (3), a class of orthogonal moments are obtained [13] and different boundary conditions have been used for the proposal of different methods with distinct definition of λ: Fourier–Bessel modes (FBM) [14], Bessel–Fourier moments (BFM) [15], and disk-harmonic coefficients (DHC) [16]. However, the main disadvantage of these methods is the lack of an explicit definition of their radial kernels other than Bessel functions. And this leads to inefficiency in terms of computation complexity.

The third direction uses harmonic functions to define R_n by taking advantage of their orthogonality. Harmonic functions are known to possess elegant orthogonality relations which are "similar in form" to the condition in Equation (3). For example, the orthogonality relation between complex exponential functions in Equation (2) "differs in form" from that in Equation (3) by a single multiplicative term r. Adaptation to overcome r was attempted and led to a number of proposals. Complex exponential functions are used in polar complex exponential transform (PCET) [17] and trigonometric functions are used in radial harmonic Fourier moments (RFHM) [18], polar cosine transform (PCT), and polar sine transform (PST) [17]. A recent study [19] gives a generic view on the use of harmonic functions to define the radial kernels, leading to the proposal of generic polar harmonic transforms (GPHT). The study also demonstrates some beneficial properties of GPHT on representation and recognition capabilities. Since harmonic functions are so much simpler than Jacobi polynomials and Bessel functions, computing GPHT thus promises to be faster than computing unit disk-based moments with radial kernels defined based on Jacobi polynomials or Bessel functions. It

should be noted that the generalization of polar harmonic transforms by introducing a parameter is similar to the generalization of the R-transform published recently [20].

Direct computation of GPHT from their definition is inefficient and may be slower than some efficient computation strategies that have been proposed for methods based on Jacobi polynomials [21], [22], [23]. This paper presents a number of novel strategies for fast computation of GPHT in order to further support them in terms of computational complexity. The proposed methods are based on the inherent recurrence relations among harmonic functions that are used in the definitions of the radial and angular kernels of GPHT. The employment of these relations leads to recursive and addition chain-based strategies for fast computation of GPHT kernels. The proposed strategies have been compared with direct computation and, in terms of kernel computation time, the best strategy is about $10\times$ faster. When compared with recursive computation of Zernike kernels using the q-recursive strategy [21], the proposed strategy is also $5\times$ faster.

The content of this paper is a comprehensive extension of a previous research work [24], which only presents recursive computation of complex exponential functions. A similar work on fast computation of polar harmonic transforms has recently presented in [25]. This work, however, focuses only on recursive computation of trigonometric functions and does not cover complex exponential functions, efficient computation flows, and addition chain-based strategy as in this paper. The next section will give some background on GPHT, their discrete approximation, and revisit a fast computation strategy based on geometrical symmetry. Sections III and IV present the recurrence relations among harmonic functions and propose some strategies for their fast computation using recursion or the shortest addition chain. Experimental results to validate the proposed strategies are then given in Section V and conclusions are finally drawn in Section VI.

II. The generic polar harmonic transforms

A generic view on the use of harmonic functions for the definition of radial kernels has recently led to the proposal of four generic classes of polar harmonic transforms (GPHT) [19]: generic polar complex exponential transform (GPCET), generic radial harmonic Fourier moment (GRHFM), generic polar cosine transform (GPCT), and generic polar sine transform (GPST). While still owning "invariance in form" and orthogonality properties, these transforms have been shown to have some other beneficial properties in terms of representation and recognition capabilities. The radial kernels of GPCET, GRHFM, GPCT, and GPST are respectively defined as

$$R_{ns}(r) = \sqrt{\frac{sr^{s-2}}{2\pi}}\, \mathrm{e}^{i2\pi n r^s}$$

$$R^H_{ns}(r) = \sqrt{\frac{sr^{s-2}}{2\pi}} \begin{cases} 1, & n = 0 \\ \sqrt{2}\sin(\pi(n+1)r^s), & n \text{ odd} > 0 \\ \sqrt{2}\cos(\pi n r^s), & n \text{ even} > 0 \end{cases}$$

$$R^C_{ns}(r) = \sqrt{\frac{sr^{s-2}}{2\pi}} \begin{cases} 1, & n = 0 \\ \sqrt{2}\cos(\pi n r^s), & n > 0 \end{cases}$$

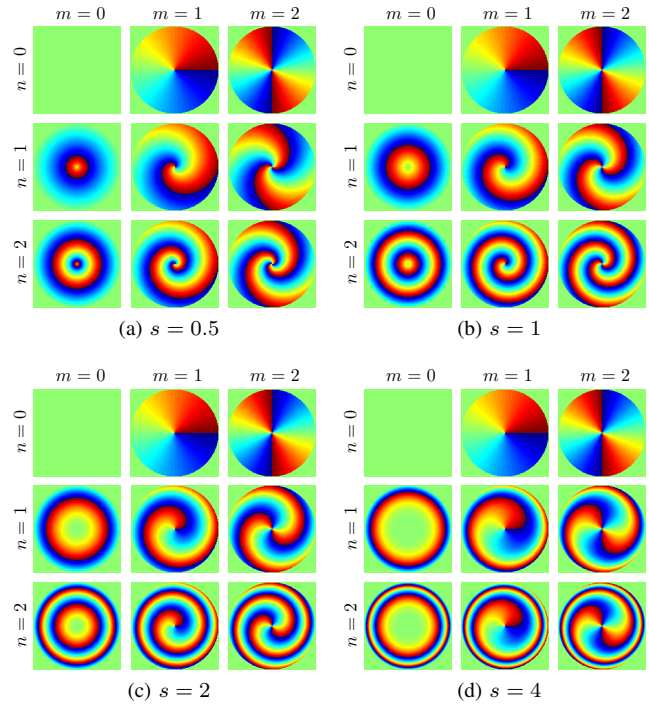


Figure 1. 2D view of the phase of GPCET kernels using $s = 0.5, 1, 2, 4$ for $\{(n, m) \in \mathbb{Z}^2 : 0 \le n, m \le 2\}$.

$$R^S_{ns}(r) = \sqrt{\frac{sr^{s-2}}{2\pi}}\, \sqrt{2}\sin(\pi n r^s), \quad n > 0.$$

It can be seen that these definitions of radial kernels satisfy the condition in Equation (3). Also, GPCET is equivalent to GRHFM in terms of representation capability, similar to the equivalence between different forms of Fourier series (namely complex exponential and trigonometric functions). In addition, the introduction of a parameter $s \in \mathbb{Z}$ [26] makes these transforms generic versions of existing polar harmonic transforms: PCET, RHFM, PCT, and PST are obtained from GPCET, RHFM, GPCT, and GPST when $s = 2, 1, 2, 2$, respectively. The two-dimensional view of the phase of some GPCET kernels is illustrated in Figure 1.

Note that all radial and angular kernels of GPHT are defined based on complex exponential and trigonometric functions. Direct computation of these functions is time-consuming and constitutes a dominant part of the computation of GPHT due to their $O(\log^2 n)$ complexity, where n is the number of precision digits at which the functions are evaluated [27]. The overall complexity may become excessively high when a large number of moments is needed, or large-sized images are used, or a high-precision computation is required. Since these requirements are common in practical applications, the existence of strategies for fast computation of GPHT kernels is vital for the applicability of GPHT. The remaining of this section will discuss the discrete approximation of GPHT and revisit a fast computation strategy based on geometrical symmetry. The next two sections (Sections III and IV) will propose some strategies to compute GPHT rapidly based on the inherent recurrence relations among harmonic functions.

Discrete approximation

The image processed in digital systems is not defined in a continuous domain, but a grid of pixels. Over each pixel region, a constant value is used to represent the intensity of the image function at that pixel location. Accordingly, the continuous integration in Equation (1) needs to be approximated in two separate ways: in the domain of integration (geometrically) and in the kernel values (numerically). To be precise, a lattice of pixels is selected to cover the unit disk for geometric approximation, and some values are assigned to each kernel at each pixel location for numerical approximation.

Geometric approximation is required to linearly map a rectangular grid of pixels onto a continuous and circular domain of the unit disk since an exact mapping is impossible due to their different geometric nature. In this work, the incircle mapping strategy [28] which puts a disk inside the image's rectangular domain as illustrated in Figure 2 is used. In this way, most of the circular region is involved in the computation. However, some image pixels that lie outside the circular region (the white pixels) will not be used in the computation. Using incircle also leads to the consideration of image pixels that lie at the disk boundary since using these pixels causes an inherent error, called *geometric error* in the literature. In the figure, the blue and cyan pixels at the disk boundary have their centers lying outside and inside the circular region, respectively. While the error caused by the cyan pixels can be fixed, that by the blue pixels cannot be fixed and remains the main source of geometric error [29]. The impact of this error type depends heavily on the behavior of the radial kernels around the point $r = 1$. Let f be the digital image of size $M \times N$, the domain $[0, M] \times [0, N]$ of f is mapped onto $[-1, 1] \times [-1, 1]$ by

$$i \longrightarrow \frac{2i - M}{M} = i\Delta x - 1, \qquad 0 \leq i \leq M$$
$$j \longrightarrow \frac{2j - N}{N} = j\Delta y - 1, \qquad 0 \leq j \leq N$$

where $\Delta x = \frac{2}{M}$ and $\Delta y = \frac{2}{N}$. In order to avoid geometric error in this work, only the image pixels that lie entirely inside the unit disk (the yellow pixels) are used in the computation. The values of the image function over the intersection regions between the unit disk and the blue/cyan pixels are then assumed to take value 0. Note that an algorithm to compute ZM that eliminates the geometric error by performing interpolation and then doing computation in polar coordinates was proposed in [30]. However, for simplicity and to avoid interpolation error, this algorithm is not used in this study.

Numerical approximation arises naturally when GPHT are computed from the approximated lattice of pixels over the unit disk. Let $[i, j]$ denote one pixel region in the domain of f having a constant intensity value $f[i, j]$, its mapped region in the unit disk is $\left[x_i - \frac{\Delta x}{2}, x_i + \frac{\Delta x}{2}\right] \times \left[y_j - \frac{\Delta y}{2}, y_j + \frac{\Delta y}{2}\right]$, where (x_i, y_j) are the coordinates of the center of the mapped region. The condition for this mapped region to be labeled yellow color is that its four corners lie inside the unit disk. Mathematically, the set $\mathcal{C}$ of pixels that satisfies this condition is defined as $\mathcal{C} = \left\{[i, j] : \left(x_i \pm \frac{\Delta x}{2}\right)^2 + \left(y_j \pm \frac{\Delta y}{2}\right)^2 \leq 1\right\}$. The integration in Equation (1) can then be approximated as a

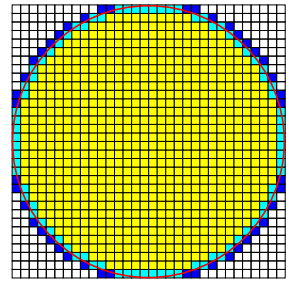


Figure 2. Lattice-point approximations of a circular region of an image of 32×32 pixels using incircle.

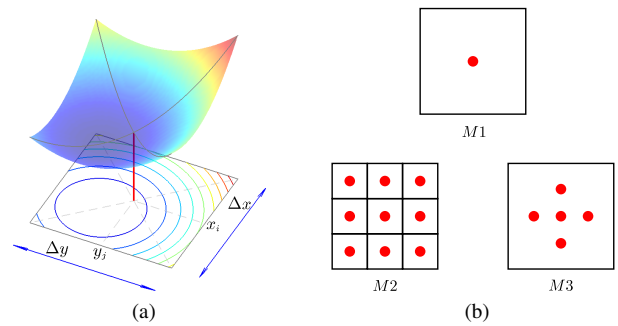


Figure 3. (a) Illustration of the value of V_{nm} over the mapped region of size $\Delta x \times \Delta y$. (b) Some approximation schemes for numerical integration of 2D functions over a square region. The red dots represent the sampling locations.

discrete sum indexed by pixels in $\mathcal{C}$:

$$\hat{H}_{nm} = \sum_{[i,j] \in \mathcal{C}} f[i, j]\, h_{nm}[i, j],$$

where $\hat{H}_{nm}$ is an estimate of H_{nm} and

$$h_{nm}[i, j] = \int_{x_i - \frac{\Delta x}{2}}^{x_i + \frac{\Delta x}{2}} \int_{y_j - \frac{\Delta y}{2}}^{y_j + \frac{\Delta y}{2}} V^*_{nm}(x, y)\, \mathrm{d}x\mathrm{d}y \tag{4}$$

represents the accumulation of V^*_{nm} over a region of size $\Delta x \times \Delta y$ in the unit disk that represents the pixel $[i, j]$ (see Figure 3a). Since V_{nm} is originally defined in polar coordinates while $h_{nm}[i, j]$ is evaluated in Cartesian ones, computing $h_{nm}[i, j]$ in practice usually relies on numerical integration techniques.

There exist many such techniques that can be employed to approximately evaluate $h_{nm}[i, j]$ [28]. The simplest is the rectangle formula (scheme $M1$ in Figure 3b) written as

$$h_{nm}[i, j] \simeq V^*_{nm}(x_i, y_j)\Delta x \Delta y. \tag{5}$$

This approximation assumes that the value of V_{nm} over the mapped region of the pixel $[i, j]$ is constant and equals its value at the central point (x_i, y_j). The order of approximation error in this case is $O\left((\Delta x \Delta y)^2\right)$, thus depends on the area of the mapped region. Therefore, a more accurate approximation could be obtained by "pseudo" up-sampling the pixel $[i, j]$ and then using Equation (5) on each up-sampled pixel (the $L = 3 \times 3$ up-sampling is shown in scheme $M2$ in Figure 3b). Other approaches use the L-dimensional cubature formulas [31] defined as

$$h_{nm}[i, j] \simeq \sum_{k=1}^{L} w_k V^*_{nm}(u_k, v_k)\Delta x \Delta y,$$

where $\{(u_k, v_k) : 1 \leq k \leq L\}$ is a set of design points belonging to the mapped region that represents the pixel $[i, j]$ and $\{w_k : 1 \leq k \leq L\}$ is a set of the corresponding weights. As an example, a five-dimensional cubature formula (scheme $M3$ in Figure 3b) has the definition $h_{nm}[i, j] \simeq \frac{1}{3}\big[-V^*_{nm}(x_i, y_j) + V^*_{nm}\big(x_i + \frac{\Delta x}{4}, y_j\big) + V^*_{nm}\big(x_i - \frac{\Delta x}{4}, y_j\big) + V^*_{nm}\big(x_i, y_j + \frac{\Delta x}{4}\big) + V^*_{nm}\big(x_i, y_j - \frac{\Delta x}{4}\big)\big]\Delta x \Delta y$. The order of approximation error now reduces to $O\big((\Delta x \Delta y)^{L+1}\big)$. Thus, a higher accuracy is obtained when a larger value of L is used. Note that the gain in accuracy has to be paid by an increase in computational complexity, which is not always preferred in real applications. In this work, for simplicity and for the purpose of comparison, the scheme $M1$ in Figure 3b is used to compute $h_{nm}[i, j]$ in GPHT and comparison methods.

In the literature, the aforementioned approximation error in computing $h_{nm}[i, j]$ is often called *numerical error* [28]. The impact of this type of error depends heavily on the behavior of V_{nm} over the mapped region that represents the pixel $[i, j]$ and thus varies according to the actual moment and the orders n, m. In the case of GPHT, the radial and angular kernels that constitute V_{nm} are defined based on harmonic functions and oscillate within their respective intervals, i.e., $[0, 1]$ and $[0, 2\pi)$. This means that V_{nm} will oscillate at a higher frequency when n and m take higher values. In addition, since the aforementioned approximation schemes are based on the sampled values of V_{nm}, the value of H_{nm} is susceptible to information loss if V_{nm} has been under-sampled. This is because the number of sampling points is fixed and determined by the size of the input image. Thus, when a moment of a too-high order n or m is considered, a fixed sampling of V_{nm} becomes insufficient.

There exist some approaches that try to eliminate numerical error by avoiding direct approximation of $h_{nm}[i, j]$ over each mapped region of the pixel $[i, j]$. Some stick to the Cartesian space and perform the integration in Equation (4) analytically after converting V_{nm} from being defined by polar coordinates to being defined by Cartesian ones [32], [33]. This is possible due to the relationship between Zernike and geometric moments [6] or by piecewise polynomial interpolation of V_{nm}. Another approach [30] computes the moments directly in polar space by using bicubic interpolation to convert the digital image f from being defined as a piecewise constant function in the Cartesian space to being defined as a piecewise constant function in the polar space. Optimization techniques were recently used to improve the orthogonality of the approximated discrete kernels for more accurate image reconstruction [34]. However, these approaches to eliminate numerical error are much more computationally expensive than the aforementioned approximation schemes because interpolation makes any recurrence relation between radial and/or angular kernels invalid and also optimization requires iterative evaluations. In addition, these approaches also introduce new error into the value of H_{nm} due to the interpolation or optimization process. For these reasons, in this work, these approaches are not used to compute GPHT and comparison methods.

Geometrical symmetry-based fast computation

It can be seen from Figure 4 that the seven symmetrical points of a point P_1 across the y-axis, the origin, the x-axis, and the line $y = x$ in the Cartesian coordinate system have the same radial and related angular coordinates as those of P_1 (see Table I). From the properties of complex exponential functions, the radial kernel takes the same value and the angular kernel has related values at these points. Thus, geometrical symmetry in the distribution of pixels inside the unit disk can be exploited to reduce the need of computing the radial and angular kernels for all pixels inside the unit disk to only pixels in one of the eight equal pies in Figure 4. This idea was initially proposed in [35] for Zernike moments and is applicable to all unit disk-based moments. In addition, due to its geometrical nature, this strategy is independent of the aforementioned pixel-based discrete approximation and thus can be used in combination with strategies for fast computation of radial/angular kernels proposed in the following two sections to further reduce the computational complexity. This observation will have experimental evidence in Section V.

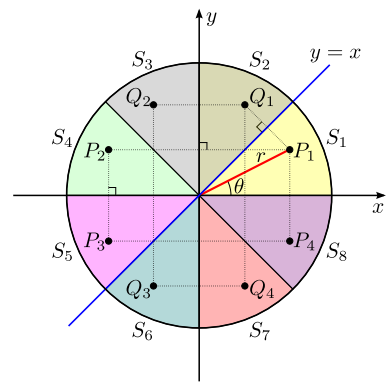


Figure 4. Symmetrical points of a point P_1 inside the unit disk across the y-axis, the origin, and the x-axis.

Table I
CARTESIAN AND POLAR COORDINATES OF THE SYMMETRICAL POINTS OF A POINT P_1 EXPRESSED VIA THOSE OF P_1.

Symmetrical point	Symmetrical axis	Cartesian coordinates	Radial coordinate	Angular coordinate
P_1		(x, y)	r	θ
P_2	y-axis	$(-x, y)$	r	$\pi - \theta$
P_3	origin	$(-x, -y)$	r	$\pi + \theta$
P_4	x-axis	$(x, -y)$	r	$-\theta$
Q_1	$y = x$	(y, x)	r	$\frac{\pi}{2} - \theta$
Q_2	$y = x$, y-axis	$(-y, x)$	r	$\frac{\pi}{2} + \theta$
Q_3	$y = x$, origin	$(-y, -x)$	r	$-\frac{\pi}{2} - \theta$
Q_4	$y = x$, x-axis	$(y, -x)$	r	$-\frac{\pi}{2} + \theta$

III. FAST COMPUTATION OF EXPONENTIAL FUNCTIONS

Complex exponential functions are used in the definition of GPCET radial kernels and GPHT angular kernels. This section presents two different strategies to compute these functions efficiently using recursion and the shortest addition chain.

Recursive exponentiation

From the following recursive definition of exponentiation:

$$\text{base case:} \quad \mathrm{e}^{0\alpha} = 1,$$

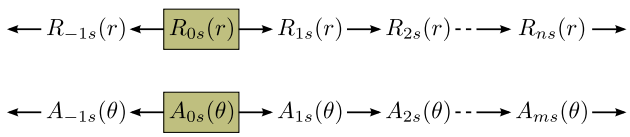


Figure 5. Computation of R_{ns} and A_m based on recursive computation of complex exponential functions.

$$\text{inductive clause:} \quad \mathrm{e}^{k\alpha} = \mathrm{e}^{(k-1)\alpha}\,\mathrm{e}^{\alpha}, \quad k \in \mathbb{Z},$$

the complex exponential functions in the definition of GPCET radial kernels, R_{ns}, and GPHT angular kernels, A_m, can be computed recursively as

$$\begin{aligned} \mathrm{e}^{i2\pi n r^s} &= \mathrm{e}^{i2\pi(n-1)r^s}\,\mathrm{e}^{i2\pi r^s}, \\ \mathrm{e}^{im\theta} &= \mathrm{e}^{i(m-1)\theta}\,\mathrm{e}^{i\theta}, \end{aligned} \tag{6}$$

using the base cases $\mathrm{e}^{i2\pi 0 r^s} = 1$ and $\mathrm{e}^{i0\theta} = 1$, respectively. Assuming that $\left\{\sqrt{\frac{sr^{s-2}}{2\pi}}, \mathrm{e}^{i2\pi r^s}, \mathrm{e}^{i\theta}\right\}$ have been pre-computed and stored in memory for the polar coordinates (r, θ) of all pixel centers mapped inside the unit disk, the following recurrence relations of R_{ns} and A_m:

$$R_{ns}(r) = \sqrt{\frac{sr^{s-2}}{2\pi}}\,\mathrm{e}^{i2\pi n r^s} = R_{(n-1)s}(r)\,\mathrm{e}^{i2\pi r^s}, \tag{7}$$

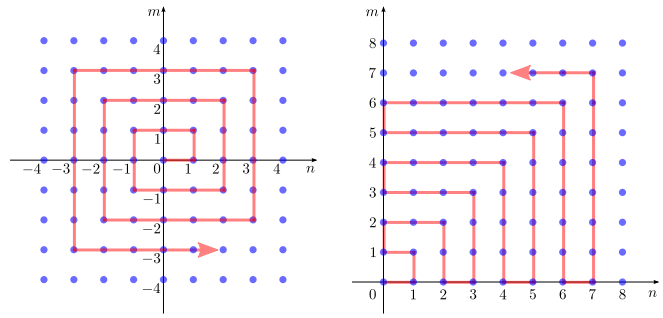
$$A_m(\theta) = \mathrm{e}^{im\theta} = A_{m-1}(\theta)\,\mathrm{e}^{i\theta}, \tag{8}$$

lead to their recursive computation with the base cases $R_{0s}(r) = \sqrt{\frac{sr^{s-2}}{2\pi}}$ and $A_0(\theta) = 1$, respectively. Thus, computing R_{ns} from $R_{(n-1)s}$ and A_m from A_{m-1} each requires only one multiplication for each pixel location, which is very fast when compared to direct exponentiation. It should be noted that these forms of recurrence relations are simpler than those that were previously proposed for the radial kernels defined based on Jacobi polynomials (see [21], [22] and references therein). By using Equations (7) and (8), only two recursive computational threads are needed to reach every GPCET radial kernel and GPHT angular kernel, whereas many threads would be required to cover all radial kernels defined based on Jacobi polynomials. The corresponding computation flows for R_{ns} and A_m are illustrated in Figure 5. Note also that the method proposed here is much faster than the one mentioned in [17] where direct exponentiation is used to compute PCET radial and angular kernels.

The aforementioned recursive computation of R_{ns} and A_m could be employed for extremely fast computation of GPCET kernels when the order set $\mathcal{S}$ composes a square region in $\mathbb{Z}^2$ and takes the origin as its center as

$$\mathcal{S} = \{(n, m) : n, m \in \mathbb{Z}^2, |n|, |m| \leq K\},$$

where K is a certain positive integer. By using the computational flow depicted in Figure 6a, computing $V_{nms}(r, \theta)$ requires only two multiplications: one for updating $R_{ns}(r)$ or $A_m(\theta)$ and one for computing $V_{nms}(r, \theta) = R_{ns}(r)A_m(\theta)$. As a result, computing a GPCET moment requires only three vector multiplications (two for computing V_{nms} and one for multiplying V_{nms} by f), followed by a discrete sum over all the pixel locations.



(a) Four quadrants (b) One quadrant

Figure 6. Computation flows for GPCET kernels starting from the order $(0, 0)$. At each step, one of the orders n and m changes its value by adding 1 or -1, depending on the current direction of the flow. Computing each kernel V_{nms} thus requires two multiplications, one for computing R_{ns} or A_m using Equations (7) or (8), depending on whether the value of n or m has just been changed, and the other for multiplying R_{ns} by A_m to get V_{nms}.

To further boost the computation speed, instead of letting the computational flow to visit all $(n, m) \in \mathcal{S}$ in the four quadrants in the Cartesian space, it is sufficient to visit only $(n, m) \in \mathcal{S}$ in one quadrant $(n, m \geq 0)$ as illustrated in Figure 6b. This is possible due to the following relations:

$$\begin{aligned} R_{-ns}(r) &= R^*_{ns}(r), \\ A_{-m}(\theta) &= A^*_m(\theta). \end{aligned}$$

Thus, whenever R_{ns} and A_m are available, computing the four related GPCET kernels requires only two multiplications and three conjugations:

$$\begin{aligned} V_{nms}(r, \theta) &= R_{ns}(r)A_m(\theta), \\ V_{-nms}(r, \theta) &= R^*_{ns}(r)A_m(\theta), \\ V_{-n-ms}(r, \theta) &= V^*_{nms}(r, \theta), \\ V_{n-ms}(r, \theta) &= V^*_{-nms}(r, \theta). \end{aligned}$$

Since eight multiplications are needed to compute these four kernels if the computational flow in Figure 6a is used, the simple trick of staying in one quadrant leads to a $\frac{8}{3}$-time reduction in the number of multiplications.

When there is enough memory space to store all the radial kernels R_{ns} $(|n| \leq K)$ and angular kernels A_m $(|m| \leq K)$, then pre-computing and keeping their values in memory may lead to a further reduction in computational complexity by using the computational flow in Figure 7. It can be seen that, for each pixel location, the total number of multiplications required to update the values of R_{ns} or A_m using the computational flow in Figure 6a is $(K + 1)^2 - 1$ whereas pre-computation using the computational flow in Figure 7 needs only $2K$ multiplications. However, besides the memory requirement to store pre-computed values, this strategy has a disadvantage concerning the data accessing time. The pre-computed values of all R_{ns} (or A_m) are often stored in a matrix $\mathbf{R}_s$ (or $\mathbf{A}$) indexed by r and n (or θ and m). When the kernel V_{nms} needs to be computed, the values in the nth column of $\mathbf{R}_s$ and in the mth column of $\mathbf{A}$ are retrieved from memory. Since fetching data from memory is less efficient than directly using data in CPU cache as in the case of recursive computation, the pre-

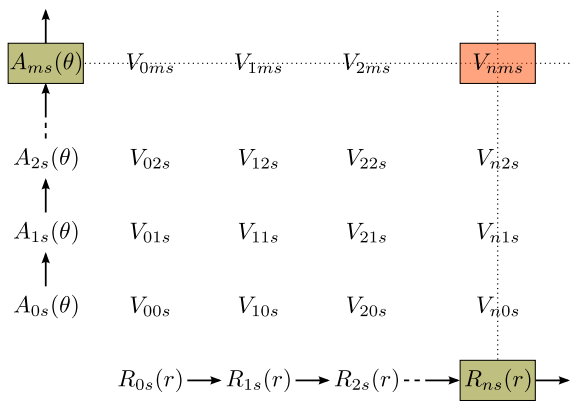


Figure 7. Computation of GPCET kernels V_{nms} from the pre-computed and stored values of the radial kernels R_{ns} and angular kernels A_m $(0 \leq n, m \leq K)$.

computing strategy may not provide any computational speed-up as expected. This concern will be supported by experimental evidence in Section V.

In real situations, the speed-up in computing GPCET kernels leads to the possibility of increasing the number of GPCET moments without changing the system throughput. This increase is equivalent to an increase in the number of features or, more importantly, an increase in the number of pattern classes the system can discriminate. Another by-product of fast computation is the reduction in storage space requirement. In methods based on orthogonal polynomials the common practice is to pre-compute and store kernels in memory in order to avoid their repetitive expensive computation. For example, if L moments are needed then the L corresponding kernels need to be pre-computed and stored. In the case of GPCET, since the kernel of any order can be computed rapidly, there is merely a need to store the set $\left\{\sqrt{\frac{sr^{s-2}}{2\pi}}, \mathrm{e}^{i2\pi r^s}, \mathrm{e}^{i\theta}\right\}$ for the polar coordinates (r, θ) of all the pixel centers mapped into the unit disk, regardless the required number of moments. This practice also leads to a reduction in storage space by $\frac{2K}{3}$.

Addition-chain-based exponentiation

In some situations where only some kernels V_{nms} of certain orders n and m are needed, the recursive strategies proposed previously in this section are not optimal since all R_{ps} and A_q with $0 \leq p \leq n$ and $0 \leq q \leq m$ need to be computed (Figure 5), leading to $n + m + 1$ multiplications for each pixel location. There exists in the literature methods that allow computing R_{ns} from R_{0s} and A_m from A_0 quickly using addition-chain exponentiation [36]. The basic idea is based on the minimal number of multiplications necessary to compute g^n from g $(n \in \mathbb{Z})$, given that the only permitted operation is multiplication of two already-computed powers of g. In fact, this minimal number is the length of the shortest addition chain for n, which is a list of positive integers $a_1, a_2, \ldots, a_\ell$ with $a_1 = 1$ and $a_\ell = n$ such that for each $1 < i \leq n$, there are some j and k satisfying $1 \leq j \leq k < i$ and $a_i = a_j + a_k$. It thus can be seen that a shorter addition chain for n results in faster computation of g^n by computing $g^{a_2}, g^{a_3}, \ldots, g^{a_{\ell-1}}, g^n$ in sequence. While the shortest addition chain for a relatively small number can be obtained from lookup table [37], finding this chain for a large number is known as an NP-complete problem [38]. For this reason, a near-optimal solution is often used as a substitution in practical situation.

Algorithm 1 Computation of g^n by the "square-and-multiply" exponentiation method.

```
a ← 1
for i = ℓ → 0 do
    a ← a * a
    if c_i = 1 then
        a = a * g
    end if
end for
```

A number of methods has been proposed to find near-optimal addition chains for a number. One of the simplest and most frequently used is the binary method, also known as the "square-and-multiply" method, which is based on the following formula to compute g^n $(n > 0)$ recursively using squaring and multiplication as

$$g^n = \begin{cases} 1, & \text{if } n = 0, \\ g \times \left(g^{\frac{n-1}{2}}\right)^2, & \text{if } n \text{ is odd} \\ \left(g^{\frac{n}{2}}\right)^2, & \text{if } n \text{ is even} \end{cases}$$

Let $n = \sum_{i=0}^{\ell} c_i\, 2^i$ be the binary expansion of a positive integer n, the pseudo-code for the computation of g^n $(n > 0)$ using the "square-and-multiply" method is given in Algorithm 1. At the beginning of each iteration i of the "for loop", a is equal to g^t, where t has its binary expansion equal to the first $\ell - i$ prefix of the binary expansion of n. Squaring a has the effect of doubling t, and multiplying by g puts a 1 in the last digit if the corresponding bit c_i is 1. It can be seen that this algorithm uses $\log n$ squaring operations and at most $\log n$ multiplications. In other words, the "square-and-multiply" method takes $2 \log n$ multiplications in the worst case and $\frac{3}{2} \log n$ on average. Since the lower bound for the number of multiplications required to do a single exponentiation is $\log n$, this method is usually considered as "good enough" and is computationally much more efficient than the recursive method of multiplying the base g with itself repeatedly to get g^n, requiring n multiplications. The application of this "square-and-multiply" method for the computation of GPCET radial kernels and GPHT angular kernels when only some moments of certain orders need to be computed will speedup the computation. This observation will be supported by experimental evidence in Section V.

Comparison with the existing recursion-based method

In order to evaluate the improvement in computational complexity obtained by the proposed method over the method recently presented in [25], a theoretical comparison on the computational complexity of PCET (or GPCET at $s = 2$) has been carried out and the results are summarized in Table II.

Table II
COMPUTATIONAL COMPLEXITY COMPARISON ON RECURSIVE COMPUTATION OF THE ANGULAR AND RADIAL KERNELS OF PCET.

	Addition	Multiplication	Table lookup
[25]	3	4	2
Proposed method	0	1	1

Note that [25] only presents recursive computation of complex exponential functions for the angular and radial kernels of PCET by employing the recurrence relations of trigonometric functions. It does not have efficient computation flows as in Figure 6 and the addition chain-based exponentiation presented in this paper. So a direct comparison with the method in [25] should only involve Equations (7), (8) and Figure 5. Nevertheless, the comparison results demonstrate clearly the superiority of the proposed method over the method in [25].

IV. FAST COMPUTATION OF TRIGONOMETRIC FUNCTIONS

The trigonometric functions used in the definition of the radial kernels of GRHFM, GPCT, and GPST can also be computed rapidly. This is because cosine and sine functions are equivalent to complex exponential functions in terms of computational complexity due to

$$\cos(\pi n r^s) = \mathrm{Re}\{\mathrm{e}^{i\pi n r^s}\} = \frac{\mathrm{e}^{i\pi n r^s} + \mathrm{e}^{-i\pi n r^s}}{2},$$
$$\sin(\pi n r^s) = \mathrm{Im}\{\mathrm{e}^{i\pi n r^s}\} = \frac{\mathrm{e}^{i\pi n r^s} - \mathrm{e}^{-i\pi n r^s}}{2i}.$$

By using these identities, the recursive strategies to compute GPCET radial kernels and GPHT angular kernels presented in Section III can also be employed to compute the radial kernels of GRHFM, GPCT, and GPST. In other words, the implementation of GRHFM, GPCT, and GPST can rely on that of GPCET for low computational complexity. In this way, all the computational gains claimed for GPCET in the previous section by using recursion and addition chain are still valid for GRHFM, GPCT, and GPST.

Recursive computation

Apart from relying on complex exponential functions, cosine and sine functions could also be fast computed by using the following recurrence relations:

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_{n+1}(x) &= 2x\,T_n(x) - T_{n-1}(x), \quad \text{for } n \geq 1, \end{aligned} \tag{9}$$

and

$$\begin{aligned} U_0(x) &= 1 \\ U_1(x) &= 2x \\ U_{n+1}(x) &= 2x\,U_n(x) - U_{n-1}(x), \quad \text{for } n \geq 1 \end{aligned} \tag{10}$$

with

$$T_n(\cos(\pi r^s)) = \cos(\pi n r^s),$$
$$U_n(\cos(\pi r^s)) = \frac{\sin(\pi(n+1)r^s)}{\sin(\pi r^s)}$$
$$\sin(\pi n r^s) = U_{n-1}(\cos(\pi r^s))\sin(\pi r^s),$$

where T_n and U_n are the Chebyshev polynomials of the first and second kinds, respectively [39]. Accordingly, the computation of $\cos(\pi n r^s)$ and $\sin(\pi n r^s)$ can also be carried out recursively with the base cases $\{1, \cos(\pi r^s)\}$ for Equation (9) and $\{1, 2\cos(\pi r^s)\}$ for Equation (10), provided that $2\cos(\pi r^s)$ and $\sin(\pi r^s)$ have been pre-computed and are available throughout the computation process. The computation flow for GRHFM, GPCT, and GPST radial kernels are illustrated in Figure 8. It can be seen that computing $\cos(\pi n r^s)$ and $\sin(\pi n r^s)$ each requires only one multiplication and one subtraction, which is almost equivalent to one multiplication required to compute $\mathrm{e}^{i2\pi n r^s}$ as in Equation (6).

Addition-chain-based computation

Fast computation of cosine and sine functions for certain n-multiple angles using the "square-and-multiply" method is also possible by using the recurrence relations of Chebyshev polynomials of the first and second kinds [40, page 782]

$$\begin{aligned} T_{n_1+n_2}(x) + T_{n_1-n_2}(x) &= 2\,T_{n_1}(x)\,T_{n_2}(x), \\ U_{2n-1}(x) &= 2\,T_n(x)\,U_{n-1}(x), \\ U_{n_1+n_2-1}(x) - U_{n_1-n_2-1}(x) &= 2\,T_{n_1}(x)\,U_{n_2-1}(x). \end{aligned}$$

By letting $n_2 = n_1 = n$ and $n_2 = n_1 - 1 = n - 1$:

$$\begin{aligned} T_{2n}(x) &= 2\,T_n^2(x) - 1, \\ T_{2n-1}(x) &= 2\,T_n(x)\,T_{n-1}(x) - x, \\ U_{2n-1}(x) &= 2\,T_n(x)\,U_{n-1}(x), \\ U_{2n-2}(x) &= 2\,T_n(x)\,U_{n-2}(x) + 1. \end{aligned}$$

These identities are "similar in form" to $g^{2n} = (g^n)^2$ and $g^{2n-1} = g\,(g^{n-1})^2$, which are the bases for the "square-and-multiply" exponentiation. It can also be seen here that using addition chain to compute $\cos(\pi n r^s)$ and $\sin(\pi n r^s)$ is computationally more efficient than using recursion.

V. EXPERIMENTAL RESULTS

The computational complexity of existing unit disk-based orthogonal moments is evaluated in terms of the elapsed time required to compute their kernels/moments from an image of size 128×128. For this image size, the incircle illustrated in Figure 2 contains 12596 pixels that lie entirely inside the unit disk (the yellow pixels). Experiments are performed on a single core of a PC equipped with a 2.33 GHz CPU, 4 GB RAM, and running Linux kernel 2.6.38. MATLAB version 7.7 (R2008b) is used as the programming environment. Let K be a certain integer constant, all the kernels/moments of orders (n, m) of each method that satisfy the conditions listed in Table III are computed and the average elapsed time over all feasible orders is used as the kernel/moment computation time at that value of K. In order to study the trends in the dependence of kernel/moment computation time on the maximal kernel/moment order K, the value of K is varied in the range $0 \leq K \leq 20$ in all experiments. In addition,

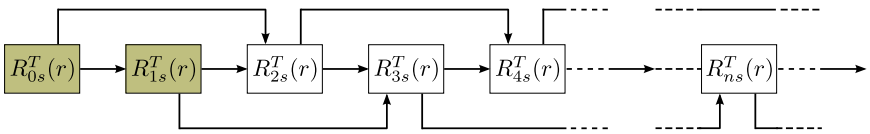


Figure 8. Computation of GRHFM, GPCT, and GPST radial kernels based on recursive computation of cosine and sine functions.

Table III
THE CONSTRAINTS ON THE MOMENT ORDERS (n, m) OF COMPARISON METHODS FOR A FIXED VALUE OF K IN THE EXPERIMENTS ON COMPUTATIONAL COMPLEXITY.

Method	Order range
ZM	$\lvert m\rvert \leq n \leq K$, $n - \lvert m\rvert =$ even
PZM	$\lvert m\rvert \leq n \leq K$
OFMM/CHFM/PJFM	$0 \leq \lvert m\rvert, n \leq K$
FBM/BFM/DHC	$0 \leq \lvert m\rvert, n \leq K$
GPCET	$\lvert m\rvert, \lvert n\rvert \leq K$
GRHFM	$\lvert m\rvert \leq K$, $0 \leq n \leq 2K$
GPCT	$0 \leq \lvert m\rvert, n \leq K$
GPST	$\lvert m\rvert \leq K$, $1 \leq n \leq K$

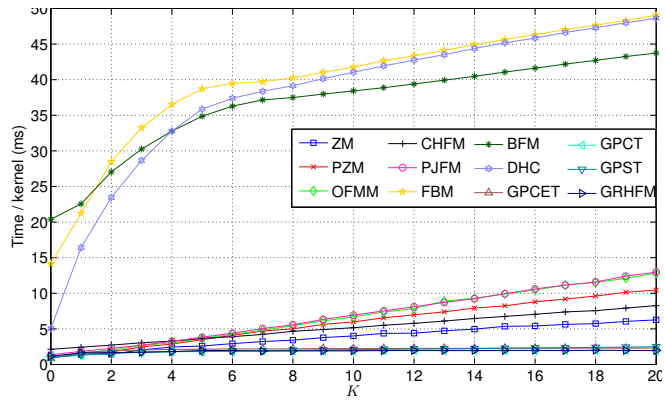


Figure 9. Kernel computation times of comparison methods by direct computation at different values of K. For each method and at a specific value of K, the kernel orders used in the computation should satisfy the conditions described in Table III and the average elapsed time over all feasible orders is used as the kernel computation time.

for more reliable results, all the computation times reported in this section are the average values over 100 trials. This section reports the computation time of kernel/moment by using definitions (Section V-A), recurrence relations (Section V-B), and the shortest addition chains (Section V-C). For fast computation of GPHT, the experiments focus on complex exponential functions (i.e., GPCET) since harmonic functions are equivalent in terms of computational complexity. This is because the recurrence relations of complex exponential and trigonometric functions developed in Sections III and IV are similar. Thus, all conclusions drawn in this section for GPCET are also valid for GRHFM, GPCT, and GPST.

A. Direct computation

In this experiment, the kernels of all unit disk-based orthogonal moments are computed using their corresponding definitions, no recursive strategy is used. The computation times per kernel in milliseconds of comparison methods at different values of K are given in Figure 9. It can be seen from the figure that Jacobi polynomial-based methods (ZM, PZM, OFMM, CHFM, PJFM) have their kernel computation times increase almost linearly with the increase in K, meaning that a longer time is needed to compute a kernel of a higher order. This is because when K increases, factorials of larger integers need to be evaluated and more additive terms in the final summation need to be computed. Among Jacobi polynomial-based methods, OFMM and PJFM have the highest and similar complexity while ZM has the lowest. This complexity ranking of these methods is consistent with the ranking according to the number of multiplications required to compute their radial kernels by using definition.

Eigenfunction-based methods (FBM, BFM, DHC) require the longest times to compute their kernels over all comparison moments. Among these methods, FBM and DHC have almost the same complexity whereas that of BFM is slightly less. The high complexity of eigenfunction-based methods can be partly explained by the need of finding the zeros of (the derivative of) Bessel functions in order to define their radial kernels. And, there exists no formulation that allows fast computation of Bessel functions in the literature. In addition, the lower complexity of BFM is due to its use of a fixed-order Bessel function while FBM and DHC use Bessel functions of different orders. In this experiment, the evaluation of Bessel functions is facilitated by the MATLAB built-in function `besselj`.

Harmonic function-based methods (GPCET, GRHFM, GPCT, GPST) each requires an almost-constant time to compute its kernels of different orders. The constant kernel computation time is the direct result of the similarity in the form of their radial kernels at different orders. A change in the kernel order only leads to a change in the input to the complex exponential function (GPCET) or cosine/sine functions (GRHFM, GPCT, GPST) and, as a result, does not affect the kernel computation time. It can also be seen from Figure 9 that the kernel computation times of harmonic function-based methods are almost the same. This is because complex exponential, cosine, and sine functions are equivalent in terms of computational complexity [27].

From the above observations on the direct kernel computation time of all unit disk-based orthogonal moments, it can be concluded here that the simple, resembling, and relating definitions of harmonic function-based kernels have resulted in an almost-constant kernel computation time, regardless of the maximal kernel order K. This makes a strong contrast with Jacobi polynomial-based and eigenfunction-based methods where a higher kernel order K means a longer kernel computation time.

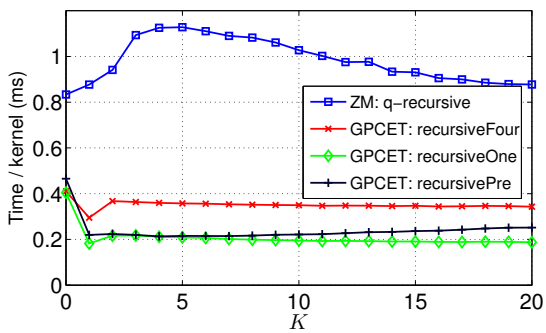


Figure 10. ZM and GPCET kernel computation times at different values of K using the q-recursive and three recursive strategies proposed in this paper.

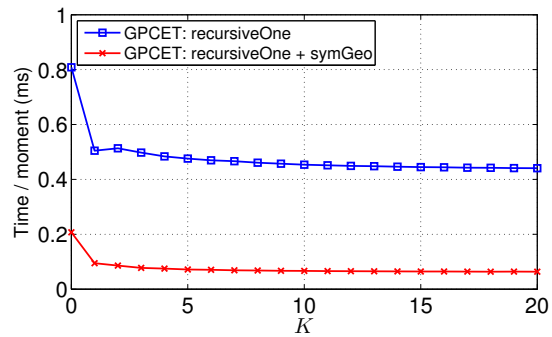


Figure 11. GPCET moment computation times at different values of K without and with geometrical symmetry.

B. Recursive computation

The proposed recursive strategies for fast computation of GPCET kernels are evaluated and compared with its direct computation and fast computation strategies for ZM kernels. The reason for comparing only to ZM lies in the lack of benchmarks on fast computation strategies for other methods and the popularity of recursive strategies for ZM computation in the literature. In this comparison, ZM kernels are computed using the q-recursive strategy [21]. The comparison results are given in Figure 10 where the legends *recursiveFour*, *recursiveOne*, and *recursivePre* denote recursive computation of GPCET kernels using the computational flows in Figures 6a, 6b, and 7, respectively. It can be seen from Figure 10 that ZM kernel computation time by using q-recursive is not constant. It increases in the range $K = 0 \rightarrow 5$ and gradually decreases when $K > 5$. This is because the q-recursive strategy, which requires pre-computing two ZM radial kernels R_{nn} and $R_{n(n-2)}$ for each order n, is applicable only when $K \geq 4$ and is profitable when $K \geq 6$. In addition, as K increases, more ZM radial kernels are computed by recursion and thus the proportion of pre-computed radial kernels decreases. This finally leads to a decrease in the average computation time of ZM radial kernels as K increases.

Using *recursiveFour* and *recursiveOne* to compute GPCET kernels leads to an almost-constant computation time, regardless of the maximal kernel order K. *recursiveOne* can be seen to be almost two-time faster than *recursiveFour*. The only exception is at $K = 1$ where the computation time suddenly drops below the average value. This might be because MATLAB simply copies the pre-computed value of $\mathrm{e}^{i\theta}$ into $A_1(\theta)$ since $A_0(\theta) = 1$, instead of performing the more complex multiplication in order to optimize the calculation. A constant computation time of GPCET kernel is due to the fact that the recurrence relations in Equations (7) and (8) do not depend on the kernel orders and that there is no need to pre-compute GPCET radial kernels of any order as in the ZM q-recursive strategy. The "purely" recursive computation of radial and angular kernels is a distinct characteristic of harmonic function-based methods. It can also be seen from the figure that the performance of *recursivePre* is not as good as expected; it is not faster than *recursiveOne*. The kernel computation time of *recursivePre*, unlike that of *recursiveOne* and *recursiveFour*, increases with the increase in K. The only possible explication for this phenomenon is the added burden of accessing column-wise data from the pre-computed matrices $\mathbf{R}_s$ and $\mathbf{A}$ as described in Section III, instead of benefiting from CPU caching. On average, recursive computation of GPCET kernels using *recursiveOne* is approximately $10\times$ faster than their direct computation and $5\times$ faster than recursive computation of ZM kernels by using the q-recursive strategy.

In computing GPCET moments, that is a combination of computing GPCET kernels and performing the inner product between these kernels and image, *recursiveOne* could be combined with the strategy based on geometrical symmetry (Section II) to further reduce the moment computation time. Figure 11 provides the average elapsed times to compute one GPCET moment using *recursiveOne* without and with geometrical symmetry (*symGeo*). It can be seen from the figure that the combination of *recursiveOne* with *symGeo* is on average $7\times$ faster than using *recursiveOne* alone, leading to an effective reduction in the computation time per moment from 0.4406 ms to 0.0634 ms at $K = 20$. These results clearly demonstrate that the strategies for fast computation of GPCET moments based on recursive computation of complex exponential functions and on geometrical symmetry are independent. Combining them definitely leads to a multiplication of the computational gains obtained individually by each of these two strategies.

C. Addition chain-based computation

Performance of fast computation of GPCET kernels using addition chain has also been evaluated at some selected values of order n. In this experiment, assuming that at each value of n, only the kernel V_{nns} is computed, not all $V_{n'm's}$ with $|n'|, |m'| \leq n$ as in the previous experiments. In the case of recursion, all the radial kernels $R_{n's}$ and angular kernels $A_{m'}$ with $n', m' \leq n$ need to be computed and the number of multiplications required to compute V_{nns} at each pixel location is $2n + 1$. On the contrary, the number of multiplications required to compute V_{nns} at each pixel location in the case of addition chain is only $2 \log n + 1$.

The comparison results between recursive and addition chain-based computation for $n = 1, 2, 4, 8, 16, 32, 64$ are given in Figure 12. It can be seen from the figure that as n increases, the GPCET kernel computation time using recursion increases almost linearly with the increase in n. This linear relationship agrees with the recursion scheme illustrated in Figure 5 where

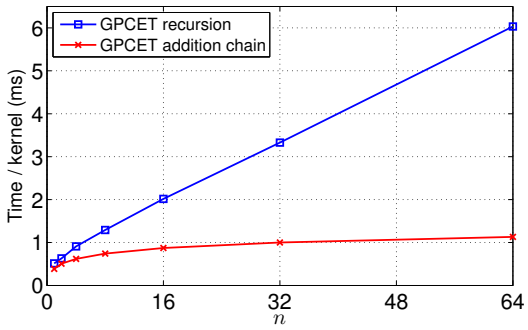


Figure 12. GPCET kernel computation times at different values of n by using recursive and addition chain-based computation.

a kernel of a higher order requires more recursion steps to compute. On the contrary, the GPCET kernel computation time using addition chain only increases logarithmically with the increase in n, agreeing with the theoretical estimation on the complexity of addition chain-based method. The benefit of using addition chain becomes more evident at a higher value of n. Thus, in situations where only harmonic kernels of certain orders need to be computed, the addition chain-based strategy should be used instead of the recursive strategy.

VI. Conclusions

Fast computation strategies for GPHT have been proposed in this paper by exploiting the recurrence relations between harmonic functions, leading to a method that is approximately $10\times$ faster than direct computation. When compared with the q-recursive strategy for fast computation of ZM kernels, the proposed method is also approximately $5\times$ faster. In addition, combination of the proposed method with the method based on geometrical symmetry leads to a multiplication of the computational gains obtained individually by the two methods. GPHT thus provide the fastest way to extract rotation-invariant features from images. Although other important issues such as robustness to discretization, noise, blurring, and symmetry preservation [41], [42] need further studies, GPHT are the promising replacements of ZM in image analysis applications where low computational complexity is an important method-selection criterion.

It is worth noting here that the value of the parameter s only slightly affects the direct computation of GPHT and does not affect at all the proposed computation strategies. This is because s has no role in the recurrence relations in Equations (7) and (8), even though it appears in the definition of GPHT radial kernels. As a result, all the conclusions drawn in this paper hold for every s. In addition, the methods proposed in this paper can also be employed to compute the angular kernels of all existing unit disk-based moments and the radial kernels of ART [43] and GFD [44] since they are defined based on harmonic functions albeit the radial kernels of ART and GFD do not satisfy the condition in Equation (3).

References

[1] M. R. Teague, "Image analysis via the general theory of moments," *Journal of the Optical Society of America*, vol. 70, no. 8, pp. 920–930, 1980.

[2] A. B. Bhatia and E. Wolf, "On the circle polynomials of Zernike and related orthogonal sets," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 50, pp. 40–48, 1954.

[3] Z. Ping, H. Ren, J. Zou, Y. Sheng, and W. Bo, "Generic orthogonal moments: Jacobi–Fourier moments for invariant image description," *Pattern Recognition*, vol. 40, no. 4, pp. 1245–1254, 2007.

[4] T. V. Hoang and S. Tabbone, "Errata and comments on "Generic orthogonal moments: Jacobi–Fourier moments for invariant image description"," *Pattern Recognition*, vol. 46, no. 11, pp. 3148–3155, 2013.

[5] A. Khotanzad and Y. H. Hong, "Invariant image recognition by Zernike moments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 5, pp. 489–497, 1990.

[6] C.-H. Teh and R. T. Chin, "On image analysis by the methods of moments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 4, pp. 496–513, 1988.

[7] Y. Sheng and L. Shen, "Orthogonal Fourier–Mellin moments for invariant pattern recognition," *Journal of the Optical Society of America A*, vol. 11, no. 6, pp. 1748–1757, 1994.

[8] Z. Ping, R. Wu, and Y. Sheng, "Image description with Chebyshev–Fourier moments," *Journal of the Optical Society of America A*, vol. 19, no. 9, pp. 1748–1754, 2002.

[9] G. Amu, S. Hasi, X. Yang, and Z. Ping, "Image analysis by pseudo-Jacobi (p = 4, q = 3)–Fourier moments," *Applied Optics*, vol. 43, no. 10, pp. 2093–2101, 2004.

[10] J. Flusser, T. Suk, and B. Zitová, *Moments and Moment Invariants in Pattern Recognition*. John Wiley & Sons, 2009.

[11] T. V. Hoang, "Image representations for pattern recognition," Ph.D. dissertation, University of Nancy, 2011.

[12] F. Bowman, *Introduction to Bessel Functions*. Dover Publications, 1958.

[13] Q. Wang, O. Ronneberger, and H. Burkhardt, "Rotational invariance based on Fourier analysis in polar and spherical coordinates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 9, pp. 1715–1722, 2009.

[14] S. Guan, C.-H. Lai, and G. W. Wei, "Fourier–Bessel analysis of patterns in a circular domain," *Physica D: Nonlinear Phenomena*, vol. 151, no. 2-4, pp. 83–98, 2001.

[15] B. Xiao, J.-F. Ma, and X. Wang, "Image analysis by Bessel–Fourier moments," *Pattern Recognition*, vol. 43, no. 8, pp. 2620–2629, 2010.

[16] S. C. Verrall and R. Kakarala, "Disk-harmonic coefficients for invariant pattern recognition," *Journal of the Optical Society of America A*, vol. 15, no. 2, pp. 389–401, 1998.

[17] P.-T. Yap, X. Jiang, and A. C. Kot, "Two-dimensional polar harmonic transforms for invariant image representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 6, pp. 1259–1270, 2010.

[18] H. Ren, Z. Ping, W. Bo, W. Wu, and Y. Sheng, "Multidistortion-invariant image recognition with radial harmonic Fourier moments," *Journal of the Optical Society of America A*, vol. 20, no. 4, pp. 631–637, 2003.

[19] T. V. Hoang and S. Tabbone, "Generic polar harmonic transforms for invariant image representation," *Image and Vision Computing*, 2014.

[20] ——, "The generalization of the R-transform for invariant pattern representation," *Pattern Recognition*, vol. 45, no. 6, pp. 2145–2163, 2012.

[21] C.-W. Chong, P. Raveendran, and R. Mukundan, "A comparative analysis of algorithms for fast computation of Zernike moments," *Pattern Recognition*, vol. 36, no. 3, pp. 731–742, 2003.

[22] C. Singh and E. Walia, "Fast and numerically stable methods for the computation of Zernike moments," *Pattern Recognition*, vol. 43, no. 7, pp. 2497–2506, 2010.

[23] W. Gautschi, *Orthogonal Polynomials: Computation and Approximation*. Oxford University Press, 2004.

[24] T. V. Hoang and S. Tabbone, "Fast computation of orthogonal polar harmonic transforms," in *Proceedings of the 21th International Conference on Pattern Recognition*, 2012, pp. 3160–3163.

[25] C. Singh and A. Kaur, "Fast computation of polar harmonic transforms," *Journal of Real-Time Image Processing*, pp. 1–8, 2012.

[26] T. V. Hoang and S. Tabbone, "Generic polar harmonic transforms for invariant image description," in *Proceedings of the 18th IEEE International Conference on Image Processing*, 2011, pp. 845–848.

[27] J. M. Borwein and P. B. Borwein, "On the complexity of familiar functions and numbers," *SIAM Review*, vol. 30, no. 4, pp. 589–601, 1988.

[28] S. X. Liao and M. Pawlak, "On the accuracy of Zernike moments for image analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1358–1364, 1998.

[29] M. Pawlak and S. X. Liao, "On the recovery of a function on a circular domain," *IEEE Transactions on Information Theory*, vol. 48, no. 10, pp. 2736–2753, 2002.

[30] Y. Xin, M. Pawlak, and S. X. Liao, "Accurate computation of Zernike moments in polar coordinates," *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 581–587, 2007.

[31] H. Engels, *Numerical Quadrature and Cubature*. Academic Press, 1980.

[32] L. G. Kotoulas and I. Andreadis, "Accurate calculation of image moments," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2028–2037, 2007.

[33] C.-Y. Wee and P. Raveendran, "On the computational aspects of Zernike moments," *Image and Vision Computing*, vol. 25, no. 6, pp. 967–980, 2007.

[34] H. Lin, J. Si, and G. P. Abousleman, "Orthogonal rotation-invariant moments for digital image processing," *IEEE Transactions on Image Processing*, vol. 17, no. 3, pp. 272–282, 2008.

[35] S.-K. Hwang and W.-Y. Kim, "A novel approach to the fast computation of Zernike moments," *Pattern Recognition*, vol. 39, no. 11, pp. 2065–2076, 2006.

[36] D. M. Gordon, "A survey of fast exponentiation methods," *Journal of Algorithms*, vol. 27, no. 1, pp. 129–146, 1998.

[37] D. E. Knuth, *The Art of Computer Programming*, 3rd ed. Addison-Wesley, 1997, vol. 2: "Seminumerical Algorithms".

[38] P. J. Downey, B. L. Leong, and R. Sethi, "Computing sequences with addition chains," *SIAM Journal on Computing*, vol. 10, no. 3, pp. 638–646, 1981.

[39] T. H. Koornwinder, R. Wong, R. Koekoek, and R. F. Swarttouw, "Orthogonal polynomials," in *NIST Handbook of Mathematical Functions*, F. W. J. Olver, D. W. Lozier, R. F. Boisvert, and C. W. Clark, Eds. Cambridge University Press, 2010, ch. 15, pp. 435–484.

[40] M. Abramowitz and I. A. Stegun, Eds., *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover Publications, 1964.

[41] M. Pawlak, *Image Analysis by Moments: Reconstruction and Computational Aspects*. Oficyna Wydawnicza Politechniki Wrocławskiej, 2006.

[42] N. Bissantz, H. Holzmann, and M. Pawlak, "Testing for image symmetries - with application to confocal microscopy," *IEEE Transactions on Information Theory*, vol. 55, no. 4, pp. 1841–1855, 2009.

[43] M. Bober, F. Preteux, and W.-Y. Y. Kim, "Shape descriptors," in *Introduction to MPEG 7: Multimedia Content Description Language*, B. S. Manjunat, P. Salembier, and T. Sikora, Eds. John Wiley & Sons, 2002, pp. 231–260.

[44] D. Zhang and G. Lu, "Shape-based image retrieval using generic Fourier descriptor," *Signal Processing: Image Communication*, vol. 17, no. 10, pp. 825–848, 2002.

Thai V. Hoang received the Diploma degree in electrical engineering from the Hanoi University of Science and Technology, Hanoi, Vietnam, in 2003, and the M.Sc. degree in robotics from the Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 2007. Since 2008, he has been a Research Assistant at LORIA, Nancy, France, and received the Ph.D. degree in computer science from Université Nancy 2, Nancy, in 2011. From 2012 to 2013, he held a post-doctoral position at Inria Nancy–Grand Est, Nancy, where he was involved in developing high-performance algorithms and software for cryoelectron microscopy. Since 2014, he has been a Post-Doctoral Researcher with the Institute of Genetics and Molecular and Cellular Biology, Illkirch, France, where he is involved in computational algorithms for whole-cell modeling using FIB/SEM data. His research interests include image processing, pattern recognition, structural bioinformatics, and high-performance computing.

Salvatore Tabbone is a Professor of Computer Science at Université de Lorraine, Nancy, France. He received the Ph.D. degree in computer science from the Institut Polytechnique de Lorraine, Nancy, in 1994. Since 2007, he has been the Head of the QGAR team at the LORIA Laboratory, Université de Lorraine, Nancy, France. His research is related to image and graphics document processing and indexing, pattern recognition, image filtering and segmentation, and content-based image retrieval. He has been the leader of several national and international projects funded by the French and European institutes. He has authored and co-authored more than 100 articles in refereed journal and conferences. He serves as a PC Member for several international and national conferences.