GPU Accelerated Polar Harmonic Transforms for Feature Extraction in ITS Applications

ZHOU YANG, MINGKAI TANG, ZHUOZHANG LI, ZILIANG REN, AND QIESHI ZHANG (Member, IEEE)

1School of Computers, Guangdong University of Technology, Guangzhou 510006, China
2CAS Key Laboratory of Human-Machine Intelligence-Synergy Systems, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China
3Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong

Corresponding author: Qieshi Zhang (qs.zhang@siat.ac.cn)

This work was supported in part by the Natural Science Foundation of China under Grant 61907009, Grant U1913202, and Grant U1813205, in part by the Science and Technology Planning Project of Guangdong Province under Grant 2017B010110007, Grant 2017B010110015, and Grant 2019B010150002, in part by the Natural Science Foundation of Guangdong Province under Grant 2019A030313802, in part by the Guangdong-Hong Kong-Macao Joint Laboratory of Human–Machine Intelligence-Synergy Systems under Grant 2019B121205007, and in part by the Shenzhen Technology Project under Grant JCYJ20180507182610734.

ABSTRACT Polar Harmonic Transform (PHT) is termed to represent a set of transforms whose kernels are basic waves and harmonic in nature, which can improve the effect in Intelligent Transportation System (ITS) applications. PHTs consist of Polar Complex Exponential Transform (PCET), Polar Cosine Transform (PCT) and Polar Sine Transform (PST). PHTs can extract orthogonal and rotation invariant features and demonstrated superior performance in ITS tasks. Farnoosh and Ali [16] use PCT coefficients to correct uncertain labels for getting more accurate body reconstruction. Al-asady and Al-amery [17] obtain accurate features for human action detection. Lin et al. [18] and Liu et al. [19] propose region duplication detection scheme for feature point mapping.

INDEX TERMS GPU, Polar harmonic transform, feature extraction, intelligent transportation system.

I. INTRODUCTION

With rapid development of artificial intelligence, intelligent transportation system (ITS) especially autonomous driving attracts multidisciplinary researchers and becomes one of most promising directions. Autonomous driving technologies are mainly divided into three parts: perceptual positioning, planning decision making and executive control. As for perceptual positioning, there are challenging tasks including driver environment understanding [1], [2], road sign detection [3], [4], pedestrian detection [5], [6], behaviour analysis and prediction [7], depth estimation [8], [9], Vehicle-to-everything (V2X) [10]–[12], and intelligent human-computer interaction technology [13], [14]. Among these ITS applications and tasks, feature extraction plays a significant role.

Polar Harmonic Transforms (PHTs) consist of Polar Complex Exponential Transform (PCET), Polar Cosine Transform (PCT) and Polar Sine Transform (PST) [15]. PHTs can extract orthogonal and rotation invariant features and demonstrated superior performance in ITS tasks. Farnoosh and Ali [16] use PCT coefficients to correct uncertain labels for getting more accurate body reconstruction. Al-asady and Al-amery [17] obtain accurate features for human action detection. Lin et al. [18] and Liu et al. [19] propose region duplication detection scheme for feature point mapping.

PHTs also show competitive result in applications like image watermarking [20]–[24], fingerprint indexing [25], image copy-move forgery detection [26]–[29], color image analysis [30], [31], breast cancer detection [32], hand vein recognition [33], binary image recognition [34], MRI data analysis [35], [36], video hashing [37], [38], airborne platform localization [39], image retrieval [40]. For compute-intensive tasks Graphics Processing Unit (GPU) based parallel computing shows obvious advantages in many fields like Wavelet transform [41], Fourier transform [42]. Computing speed is very important for ITS applications.

This paper focuses on GPU based Polar Harmonic Transforms (GPHTs) that consist of GPU based Polar Complex Exponential Transform (GPCT), Polar Cosine Transform (GPCT) and Polar Sine Transform (GPST). For utilizing
GPU parallel computational capability, we implement proposed methods with Compute Unified Device Architecture (CUDA) [43], [44]. Mathematical properties of PHTs are also considered in CUDA. Different execution configurations of GPU lead to different running time. Optimal parameter selection are evaluated as well.

The organization of this paper is as follows. The mathematics definitions of PCET, PCT and PST are given in Section 2. The proposed methods are presented in Section 3 after introducing GPU memory structure and parallel execution model. In Section 4, the performance of proposed method is evaluated as well. Finally, concludes this study.

II. POLAR HARMONIC TRANSFORMS
This section introduces PHTs, and for further details refer to [15].

A. POLAR COMPLEX EXPONENTIAL TRANSFORM (PCET)
Given a 2D image function \( f(x, y) \), it can be transformed from Cartesian coordinate to polar coordinate \( f(r, \theta) \) as following formulae transform, where \( r \) and \( \theta \) denote radius and azimuth respectively.

\[
r = \sqrt{x^2 + y^2},
\]

and

\[
\theta = \arctan \left( \frac{y}{x} \right).
\]

PCET is defined on the unit circle that \( r \leq 1 \), and can be expanded with respect to the basis functions \( H_n(r, \theta) \) as

\[
f(r, \theta) = \sum_{n=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \mathcal{M}_{nl} H_n^C(r, \theta),
\]

where the coefficient is

\[
\mathcal{M}_{nl} = \frac{1}{\pi} \int_{0}^{2\pi} \int_{0}^{1} f(r, \theta) H_n^C(r, \theta) r dr d\theta.
\]

The basis function is given by

\[
H_n(r, \theta) = R_n(r) e^{i\theta},
\]

where

\[
R_n(r) = e^{i2\pi nr^2}.
\]

Rewrite Eq. (4) with Eqs. (5) and (6):

\[
\mathcal{M}_{nl} = \frac{1}{\pi} \int_{0}^{2\pi} \int_{0}^{1} f(r, \theta) (\cos(2\pi nr^2 + i\theta) - i \sin(2\pi nr^2 + i\theta)) r dr d\theta.
\]

\(|\mathcal{M}_{nl}| \) is rotation invariant and can be used for feature extraction.

B. POLAR COSINE TRANSFORM AND POLAR SINE TRANSFORM (PCT & PST)
PCT is given by

\[
f(r, \theta) = \sum_{n=0}^{\infty} \sum_{l=-\infty}^{\infty} \mathcal{M}_{nl}^C H_n^C(r, \theta),
\]

where the coefficient is

\[
\mathcal{M}_{nl}^C = \Omega_n \int_{0}^{2\pi} \int_{0}^{1} f(r, \theta) H_n^C(r, \theta) r dr d\theta.
\]

The basis function of PCT is

\[
H_n^C(r, \theta) = R_n^C(r) e^{i\theta},
\]

where

\[
R_n^C(r) = \cos(\pi nr^2),
\]

and

\[
\Omega_n = \begin{cases} 
\frac{1}{\pi} & \text{if } n = 0 \\
\frac{2}{\pi} & \text{if } n \neq 0.
\end{cases}
\]

Rewrite Eq. (9) with Eqs. (10), (11) and (12):

\[
\mathcal{M}_{nl}^C = \Omega_n \int_{0}^{2\pi} \int_{0}^{1} f(r, \theta) \cos(\pi nr^2) \times (\cos(i\theta) - i \sin(i\theta)) r dr d\theta.
\]

Similarly, PST is given by

\[
f(r, \theta) = \sum_{n=1}^{\infty} \sum_{l=-\infty}^{\infty} \mathcal{M}_{nl}^S H_n^S(r, \theta),
\]

where the coefficient is

\[
\mathcal{M}_{nl}^S = \Omega_n \int_{0}^{2\pi} \int_{0}^{1} f(r, \theta) H_n^S(r, \theta) r dr d\theta.
\]

The basis function of PST is

\[
H_n^S(r, \theta) = R_n^S(r) e^{i\theta},
\]

where

\[
R_n^S(r) = \sin(\pi nr^2),
\]

Rewrite Eq. (15) with Eqs. (16) and (17):

\[
\mathcal{M}_{nl}^S = \Omega_n \int_{0}^{2\pi} \int_{0}^{1} f(r, \theta) \sin(\pi nr^2) \times (\cos(i\theta) - i \sin(i\theta)) r dr d\theta.
\]

PCT and PST are defined on unit circle as well. \( |\mathcal{M}_{nl}^C| \) and \( |\mathcal{M}_{nl}^S| \) are rotation invariant.
III. PROPOSED METHOD
A. GPU ARCHITECTURE
GPU contains Streaming Multiprocessors (SM). Parallel threads on SM are grouped into a thread block. Thread blocks are grouped into a grid that corresponds to a CUDA kernel function call in a GPU program [43]. Each block in a grid has its own block id. The number of threads in a thread block and the number of thread blocks in a thread grid can be specified and can also impact the computation efficiency [44].

GPU has several memory models such as register, local memory, shared memory, global memory, constant memory and can also impact the computation efficiency [44].

Logical view of GPU.

B. DIRECT COMPUTATION METHOD ON GPU
Each GPU thread handles a pixel data. Parallel threads significantly boost PHTs. Following variables can be set on threadId, x, and texture memory as shown in FIGURE 1.

For an image with \( N \times N \) resolution, each pixel is represented as \((x, y)\). The pixel in \( x\)-th column and \( y\)-th row can be mapped to GPU thread \( id \) by following equation:

\[
y \times N + x = id,
\]

where \( 0 \leq y < N, 0 \leq x < N \). \( x \) and \( y \) can be calculated from \( id \):

\[
y = \left\lfloor \frac{id}{N} \right\rfloor, \tag{21}
\]
\[
x = \text{mod}(id, N), \tag{22}
\]

where \( \text{mod}(\cdot) \) is modulo operator. According to Eqs. (21) and (22), parallel GPU threads with different \( id \) can access different \((x, y)\) pixel data to accomplish PHTs on GPU directly.

C. FAST COMPUTATION METHOD ON GPU
From Eq. (13), we can find for the pixels with same \( r \) and \( \cos(\pi mx^2) \), the different integrated part is

\[
f(r, \theta)(\cos(\theta) - i \sin(\theta)). \tag{30}
\]

Axis symmetric and origin symmetric pixels like \((x, y), (-x, y), (x, -y), (-x, -y), (y, x), (-y, x), (y, -x), (-y, -x)\) can be grouped and share computation as shown in FIGURE 2. Their Cartesian and polar coordinates are shown in Table 1.

As known \( \sin(\theta) \) and \( \cos(\theta) \) functions are periodic functions with period \( 2\pi \). Periods for \( \sin(\theta) \) and \( \cos(\theta) \) are \( 2\pi / l \). Derived from the periodic and symmetric properties of trigonometric functions that used in Fast Fourier Transform (FFT) [45], mathematical relationships for trigonometric functions exist with respect to different \( l \). If \( l \) is divided by 4 with remainder 1 that means \( \text{mod}(l, 4) = 1 \), following relationship for sine function can be deduced:

\[
\sin(l(\frac{\pi}{2} - \theta)) = \cos(\theta), \tag{23}
\]
\[
\sin(l(\frac{\pi}{2} + \theta)) = \cos(\theta), \tag{24}
\]
\[
\sin(l(\pi - \theta)) = \sin(\theta), \tag{25}
\]
\[
\sin(l(\pi + \theta)) = -\sin(\theta), \tag{26}
\]
\[
\sin(l(\frac{3\pi}{2} - \theta)) = -\cos(\theta), \tag{27}
\]
\[
\sin(l(\frac{3\pi}{2} + \theta)) = -\cos(\theta), \tag{28}
\]
\[
\sin(l(2\pi - \theta)) = -\sin(\theta). \tag{29}
\]

 Similar relationships also exist for cosine function and other \( l \) values. For the eight symmetric points on the same radius \( r \), coefficients can be calculated simultaneously.

Based on previous discussion, we rewrite Eq. (13) and have GPCT

\[
GPUM_{nl}^C = \Omega_n \iint_D w(x, y) \cos(\pi n(x^2 + y^2)) \times (G_l(x, y) - iH_l(x, y))\, dx\, dy, \tag{30}
\]

where

\[
D = \{(x, y)|0 \leq x \leq 1, 0 \leq y \leq x, 0 \leq x^2 + y^2 \leq 1\}. \tag{31}
\]
where $G_1(x, y)$ and $H_1(x, y)$ is shown in Eqs. (32) and (33), as shown at the bottom of this page, and $w(x, y)$ is given by

$$w(x, y) = \begin{cases} 
1 & \text{if } (x, y) \notin P \\
\frac{1}{2} & \text{if } (x, y) \in P, 
\end{cases}$$

(34)

where

$$P = \{(x, y)|y = x, \ -x, \ x = 0, \ y = 0\}.$$  

(35)

Similarly, GPST is given by

$$\text{GPU}M_{nl} = \Omega_o \int_D w(x, y) \sin(\pi n (x^2 + y^2)) \times (G_1(x, y) - i H_1(x, y)) \, dx \, dy.$$  

(36)

As for PCET, we can simplify rewrite Eq. (7) based mathematical property of trigonometric functions as,

$$\cos(2\pi nr^2 + l\theta) = \cos(2\pi nr^2) \cos(l\theta) - \sin(2\pi nr^2) \sin(l\theta).$$

(37)

$\Gamma_1(x, y) = \begin{cases} 
(f(x, y) + f(y, x) + f(-y, x) + f(-x, y) \\
+ f(-x, -y) + f(-y, -x) + f(y, -x) + f(x, -y) \cos(l\theta) \\
+ f(y, x) - f(-y, x) - f(-y, -x) + f(y, -x) \sin(l\theta)) & \text{if } \text{mod}(l, 4) = 0 \\
(f(x, y) - f(-x, y) - f(-x, -y) + f(x, -y) \cos(l\theta) \\
+ f(y, x) - f(-y, x) - f(-y, -x) + f(y, -x) \sin(l\theta)) & \text{if } \text{mod}(l, 4) = 1 \\
(f(x, y) - f(-x, y) - f(-x, -y) + f(x, -y) \cos(l\theta) \\
- f(y, x) + f(-y, x) - f(-y, -x) + f(y, -x) \sin(l\theta)) & \text{if } \text{mod}(l, 4) = 2, \\
(f(x, y) - f(-x, y) - f(-x, -y) + f(x, -y) \cos(l\theta) \\
- f(y, x) + f(-y, x) - f(-y, -x) + f(y, -x) \sin(l\theta)) & \text{if } \text{mod}(l, 4) = 3, 
\end{cases}$$

(32)

$H_1(x, y) = \begin{cases} 
(f(x, y) - f(y, x) + f(-y, x) - f(-x, y) \\
+ f(-x, -y) - f(-y, -x) + f(y, -x) - f(x, -y) \cos(l\theta) \\
+ f(y, x) + f(-y, x) - f(-y, -x) - f(y, -x) \sin(l\theta)) & \text{if } \text{mod}(l, 4) = 0 \\
(f(x, y) + f(-x, y) - f(-x, -y) - f(x, -y) \sin(l\theta) \\
+ f(y, x) + f(-y, x) - f(-y, -x) - f(y, -x) \cos(l\theta)) & \text{if } \text{mod}(l, 4) = 1 \\
(f(x, y) + f(-x, y) - f(-x, -y) - f(x, -y) \sin(l\theta) \\
+ f(y, x) + f(-y, x) - f(-y, -x) - f(y, -x) \cos(l\theta)) & \text{if } \text{mod}(l, 4) = 2, \\
(f(x, y) + f(-x, y) - f(-x, -y) - f(x, -y) \sin(l\theta) \\
- f(y, x) + f(-y, x) - f(-y, -x) - f(y, -x) \cos(l\theta)) & \text{if } \text{mod}(l, 4) = 3, 
\end{cases}$$

(33)

and

$$\sin(2\pi nr^2 + l\theta) = \sin(2\pi nr^2) \cos(l\theta) + \cos(2\pi nr^2) \sin(l\theta).$$

(38)

Finally, the GPCET is given by

$$\text{GPU}M_{nl} = \frac{1}{\pi} \int_D w(x, y) \times \left(\cos(2\pi n(x^2 + y^2))G_1(x, y) \\
- \sin(2\pi n(x^2 + y^2))H_1(x, y) \\
- i(\sin(2\pi n(x^2 + y^2))G_1(x, y) \\
+ \cos(2\pi n(x^2 + y^2))H_1(x, y)\right) \, dx \, dy.$$  

(39)
By using Eqs. (30), (36) and (39), a group of symmetric pixels can be handled by a GPU thread. In this case, Eqs. (21) and (22) should be reevaluated. As shown in FIGURE 3, we select one pixel from row 0, two pixels from row 1 and so on, then concatenate them for GPU threads. We have:

$$y \times \frac{(y + 1)}{2} + (N - 1 - x) = id,$$  \hspace{1cm} (40)

where $0 \leq x \leq y < n$. GPU thread $id$ is in following range

$$\frac{y \times (y + 1)}{2} < id \leq \frac{(y + 1) \times ((y + 1) + 1)}{2},$$  \hspace{1cm} (41)

$y$ can be deduced from $id$ as

$$y = \left\lfloor -1 + \sqrt{1 + 8 \times id} \right\rfloor,$$  \hspace{1cm} (42)
TABLE 4. Running time of GPCET with different unrollNum and blockDim.x.

| blockDim.x | 1     | 2     | 4     | 8     | 16    | 32    | 64    | 128   | 256   |
|------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 1          | 23.388| 20.378| 19.090| 18.492| 18.167| 18.197| 18.625| 19.518| 21.263|
| 2          | 12.309| 10.689| 9.9340| 9.6051| 9.4964| 9.6478| 10.089| 11.097| 11.348|
| 4          | 6.353 | 5.606 | 5.261 | 5.145 | 5.154 | 5.374 | 5.991 | 6.610 | 10.385|
| 8          | 3.576 | 3.144 | 2.981 | 2.959 | 3.058 | 3.930 | 4.669 | 6.223 | 10.315|
| 16         | 2.226 | 2.063 | 2.057 | 2.156 | 2.964 | 3.830 | 4.166 | 6.059 | 10.317|
| 32         | 1.888 | 1.844 | 1.858 | 2.393 | 3.463 | 3.558 | 4.177 | 6.079 | 10.339|
| 64         | 1.888 | 1.821 | 1.902 | 3.321 | 3.471 | 3.634 | 4.261 | 5.850 | 10.378|
| 128        | 1.881 | 1.820 | 1.879 | 3.436 | 3.619 | 3.669 | 3.775 | 5.541 | 10.455|
| 256        | 2.176 | 1.900 | 1.911 | 3.649 | 3.576 | 3.179 | 3.272 | 5.578 | 10.523|

D. UNROLL OPERATION

Unroll operation is an important technique that optimizes GPU execution speed by reducing branch penalties and hiding latencies including the delay from reading data [44].

In proposed method, unroll operation is to calculate more than one group of pixels in a thread. Let unrollNum be the number of groups calculated in a thread. To choose an optimal unrollNum depends on algorithm complexity and GPU memory limitation.

IV. EXPERIMENTAL RESULTS

Images with different resolution and content are tested to illustrate the feasibility and efficiency of proposed GPHTs.
FIGURE 10. Real images in ITS applications.

FIGURE 11. Running time of GPHTs with different unrollNum on real images.

FIGURE 12. Running time of GPHTs with different blockDim.x on real images.
Synthetic images are used. They are generated by following formula:

\[ f(i, j) = \text{rand}(1, 255), \quad 0 \leq i < N, \quad 0 \leq j < N, \]  

(44)

where \( \text{rand}(\cdot) \) is a function to randomly generate a integer. 1,000 synthetic images are generated. In this experiment, \( n \) ranges from 11 to 16 and \( l \) ranges from 11 to 15.

With different \( unrollNum \) and \( \text{blockDim}.x \), the running time of GPCT, GPST and GPCET are different as shown in Tables 2, 3 and 4 respectively.

\( unrollNum \) is defined as a power of two, like 2, 4, 8. For 1000 images with \( 2048 \times 2048 \) resolution, FIGURES 4, 5 and 6 show running time curve of GPCT, GPST and GPCET for different \( unrollNum \). When \( \log(unrollNum) \) is 1, GPCT, GPST and GPCET achieve the best performance.

With optimal \( unrollNum \) and \( \text{blockDim}.x \), we evaluate the impact of \( \text{blockDim}.x \). \( \text{blockDim}.x \) is defined as a power of two, like 2, 4, 8. FIGURES 7, 8 and 9 show running time curve of GPCT, GPST and GPCET for 1000 synthetic images. As for GPCT when \( \log(blockDim.x) \) is 7, proposed method is the fastest and is about 11.39 times comparing to \( \log(blockDim.x) \) is 0. Similarly, GPST and GPCET achieve the best performance when \( \log(blockDim.x) \) is 7.

With optimal \( unrollNum \) and \( \text{blockDim}.x \), we evaluate GPHTs against PHTs as shown in Table 5. While image resolution increasing, the proposed GPHTs outperform obviously. In our experiment for images with \( 2048 \times 2048 \) resolution, GPCT runs 1887.5 times faster than PCT on CPU. GPST can achieve 1795.4 times faster than PST on CPU. GPCET is 1527.1 times faster than PCET on CPU.

A. SYNTHETIC IMAGES

Sythetic images are used. They are generated by following formula:

\[
f(i, j) = \text{rand}(1, 255), \quad 0 \leq i < N, \quad 0 \leq j < N,
\]

where \( \text{rand}(\cdot) \) is a function to randomly generate an integer. 1,000 synthetic images are generated. In this experiment, \( n \) ranges from 11 to 16 and \( l \) ranges from 11 to 15.

With different \( unrollNum \) and \( \text{blockDim}.x \), the running time of GPCT, GPST and GPCET are different as shown in Tables 2, 3 and 4 respectively.

\( unrollNum \) is defined as a power of two, like 2, 4, 8. For 1000 images with \( 2048 \times 2048 \) resolution, FIGURES 4, 5 and 6 show running time curve of GPCT, GPST and GPCET for different \( unrollNum \). When \( \log(unrollNum) \) is 1, GPCT, GPST and GPCET achieve the best performance.

With optimal \( unrollNum \) and \( \text{blockDim}.x \), we evaluate the impact of \( \text{blockDim}.x \). \( \text{blockDim}.x \) is defined as a power of two, like 2, 4, 8. FIGURES 7, 8 and 9 show running time curve of GPCT, GPST and GPCET for 1000 synthetic images. As for GPCT when \( \log(blockDim.x) \) is 7, proposed method is the fastest and is about 11.39 times comparing to \( \log(blockDim.x) \) is 0. Similarly, GPST and GPCET achieve the best performance when \( \log(blockDim.x) \) is 7.

With optimal \( unrollNum \) and \( \text{blockDim}.x \), we evaluate GPHTs against PHTs as shown in Table 5. While image resolution increasing, the proposed GPHTs outperform obviously. In our experiment for images with \( 2048 \times 2048 \) resolution, GPCT runs 1887.5 times faster than PCT on CPU. GPST can achieve 1795.4 times faster than PST on CPU. GPCET is 1527.1 times faster than PCET on CPU.

B. REAL IMAGES

ITS real images are shown in FIGURE 10. 128 image patches with \( 512 \times 512 \) resolution are selected. In this experiment, \( n \) ranges from 1 to 25 and \( l \) ranges from 1 to 25.

As shown in FIGURE 11 when \( \log(unrollNum) \) is 7, GPCT, GPST and GPCET achieve the best performance. We also evaluate optimal \( \text{blockDim}.x \) as shown in FIGURE 12. When \( \log(blockDim.x) \) is 7, GPCT, GPST and GPCET achieve the best performance.

With optimal \( unrollNum \) and \( \text{blockDim}.x \), Table 6 shows the running time comparison of GPHTs and PHTs. In our experiment for real images with \( 512 \times 512 \) resolution, GPCT runs 961.8 times faster than PCT on CPU. GPST can achieve 975.3 times faster than PST on CPU. GPCET is 790 times faster than PCET on CPU.

V. CONCLUSION

In this paper, we propose GPU based PHT. By using the symmetric properties and mathematical properties of trigonometric functions, parallel GPU threads can manipulate pixels simultaneously. Formulas between GPU thread \( id \) and image pixel \((x, y)\) are deduced. For real time systems and large multimedia databases, proposed method can fully unleash GPU parallel computational capability. Comprehensive experiments are also given to illustrate the effectiveness of proposed method. Wide range of emerging applications that using PHTs will be inspired from this study.

REFERENCES

[1] W.-L. Zheng, K. Gao, G. Li, W. Liu, C. Liu, J.-Q. Liu, G. Wang, and B.-L. Lu, “Vigilance estimation using a wearable EOG device in real driving environment,” IEEE Trans. Intell. Transp. Syst., vol. 21, no. 1, pp. 170–184, Jan. 2020.

[2] T. Deng, K. Yang, Y. Li, and H. Yan, “Where does the driver look? Top-Down-Based saliency detection in a traffic driving environment,” IEEE Trans. Intell. Transp. Syst., vol. 17, no. 7, pp. 2051–2062, Jul. 2016.

[3] Q. Zhang and S.-I. Kamata, “Improved color barycenter model and its separation for road sign detection,” IEICE Trans. Inf. Syst., vol. E96.D, no. 12, pp. 2839–2849, 2013.

[4] Q. Zhang and S.-I. Kamata, “A novel color descriptor for road-sign detection,” IEICE Trans. Fundam. Electron., Commun. Comput. Sci., vol. E96.A, no. 5, pp. 971–979, 2013.

[5] N. K. Ragesh and R. Rajesh, “Pedestrian detection in automotive safety: Understanding state-of-the-art,” IEEE Access, vol. 7, pp. 47864–47890, 2019.

---

**TABLE 5.** Running time of PHTs and GPHTs on synthetic images.

| Resolution | Transform | CPU(s) | GPU(s) | CPU/GPU |
|------------|-----------|--------|--------|---------|
| 256 × 256  | PCT       | 122.71 | 0.581  | 213.30  |
| 256 × 256  | PST       | 122.78 | 0.570  | 215.33  |
| 256 × 256  | PCET      | 88.15  | 0.573  | 153.72  |
| 512 × 512  | PCT       | 577.29 | 0.688  | 838.51  |
| 512 × 512  | PST       | 553.15 | 0.689  | 801.77  |
| 512 × 512  | PCET      | 473.40 | 0.735  | 644.34  |
| 1024 × 1024| PCT       | 2640.70| 1.903  | 1387.40 |
| 1024 × 1024| PST       | 2429.90| 1.761  | 1493.10 |
| 1024 × 1024| PCET      | 2113.40| 1.820  | 1161.20 |
| 2048 × 2048| PCT       | 10808.00| 5.726  | 1887.50 |
| 2048 × 2048| PST       | 10187.00| 5.674  | 1795.40 |
| 2048 × 2048| PCET      | 8772.40| 5.745  | 1527.10 |

**TABLE 6.** Running time of PHTs and GPHTs on real images.

| Resolution | Transform | CPU(s) | GPU(s) | CPU/GPU |
|------------|-----------|--------|--------|---------|
| 512 × 512  | PCT       | 1610.4 | 1.6742 | 961.89  |
| 512 × 512  | PST       | 1628.0 | 1.6691 | 975.38  |
| 512 × 512  | PCET      | 1353.8 | 1.7136 | 790.03  |

Windows 10 and Visual Studio 2015 are used to perform experiments. CPU (Intel Core i3-8100) has 4 cores with 3.6GHz frequency, GPU (Nvidia GeForce GTX 1060) has 1280 cores with 1594MHz frequency and graphic memory is 6GB, CUDA version is 9.0.176.
J. Zhang, X. Hu, Z. Ning, E. C.-H. Ngai, L. Zhou, J. Wei, J. Cheng, P.-T. Yap, X. Jiang, and A. C. Kot, “Two-dimensional polar harmonic transforms for invariant image representation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, no. 7, pp. 1259–1270, Jul. 2010.

Z. Yang, G. Chen, and D. Shen, “Robust construction of diffusion MRI atlases with correction for inter-subject fiber dispersion,” in Computational Diffusion MRI. VA, USA: Springer, 2017, pp. 113–121.

Z. Yang, G. Chen, D. Shen, and P.-T. Yap, “Robust fusion of MRI data for template construction,” Sci. Rep., vol. 7, no. 1, Dec. 2017, Art. no. 12950.

H. Chen, Y. Wo, and G. Han, “Multi-granularity geometrically robust video hashing for tampering detection,” Multimedia Tools Appl., vol. 77, no. 5, pp. 5303–5321, Mar. 2018.

Z. Yang, G. Chen, D. Shen, and P.-T. Yap, “Hand vein recognition based on PCET,” Optik, vol. 127, no. 19, pp. 7663–7669, Oct. 2016.

J. Wu, S. Qiu, Y. Kong, Y. Chen, L. Senhadji, and H. Shu, “MomentsNet: A simple learning-free method for binary image recognition,” in Proc. IEEE Int. Conf. Image Process. (ICIP), Sep. 2017, pp. 2667–2671.

Z. Yang, G. Chen, and D. Shen, “Depth estimation for mobile edge computing networks,” IEEE Trans. Ind. Informat., vol. 16, no. 2, pp. 1352–1361, Feb. 2020.

Z. Yang, G. Wang, M. Li, and W. Du, “Hand vein recognition based on PCET,” Optik, vol. 127, no. 19, pp. 7663–7669, Oct. 2016.

J. Wu, S. Qiu, Y. Kong, Y. Chen, L. Senhadji, and H. Shu, “MomentsNet: A simple learning-free method for binary image recognition,” in Proc. IEEE Int. Conf. Image Process. (ICIP), Sep. 2017, pp. 2667–2671.
MINGKAI TANG is currently pursuing the bachelor’s degree with the Guangdong University of Technology. His main research interests include image processing and computer vision.

ZHOUZHANG LI is currently pursuing the bachelor’s degree with the Guangdong University of Technology. His main research interests include image processing and artificial intelligence.

ZILIANG REN received the Ph.D. degree from the South China University of Technology, Guangzhou, China, in 2017. He is currently holding a postdoctoral position and an Assistant Researcher with the Guangdong Provincial Key Laboratory of Robotics and Intelligent System, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. His current research interests include computer vision and machine learning.

QIESHI ZHANG (Member, IEEE) received the Ph.D. degree from Waseda University, Japan, in 2014. From 2012 to 2016, he was a Research Assistant and a Research Associate with the Information, Production, and Systems Research Center, Waseda University. He is currently an Associate Professor with the Guangdong Provincial Key Laboratory of Robotics and Intelligent System, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China. He has authored or coauthored over 70 scientific articles in international journals and conferences. His current research interests include computer vision, autonomous driving, and intelligent robots. He serves as the technical/program committee member for over 50 conferences and over 100 times. From 2010 to 2012, he was a Research Fellow with the Japan Society for the Promotion of Science (JSPS), Japan.

* * *