High-dimensional data compression via PHLCT

Zhihua Zhang and Naoki Saito

Department of Mathematics, University of California, Davis, CA 95616 USA

ABSTRACT

The polyharmonic local cosine transform (PHLCT), presented by Yamatani and Saito[1] in 2006, is a new tool for local image analysis and synthesis. It can compress and decompress images with better visual fidelity, less blocking artifacts, and better PSNR than those processed by the JPEG-DCT algorithm. Now, we generalize PHLCT to the high-dimensional case and apply it to compress the high-dimensional data. For this purpose, we give the solution of the high-dimensional Poisson equation with the Neumann boundary condition. In order to reduce the number of coefficients of PHLCT, we use not only d-dimensional PHLCT decomposition, but also $d-1, d-2, \dots, 1$ dimensional PHLCT decompositions. We find that our algorithm can more efficiently compress the high-dimensional data than the block DCT algorithm. We will demonstrate our claim using both synthetic and real 3D datasets.

Keywords: Discrete Cosine Transform, Polyharmonic local cosine transform, High-dimensional data, Compression, Poisson's equation

1. INTRODUCTION

For a periodic smooth function, the rate of the decay of its Fourier coefficients depends on the smoothness of this function. However, for a non-periodic smooth function, if, after brute-force segmentation, we extend it to a periodic function, then we obtain a periodic function that is discontinuous at the boundary. Hence, its Fourier coefficients decay very slow. To solve this problem, one presents the Discrete Cosine Transform (DCT)[5]: Let us consider a smooth function defined on a square. We extend the function by "even" reflection at the boundary. This even extension of a function is continuous across the square boundaries. Afterwards, we expand the even extension of the function into the Fourier cosine series. DCT is just the discrete version of Fourier cosine series. However, after even extension, the function is not a continuously differentiable function across the square boundaries. Hence, DCT coefficients decay still slowly. In 2006, Saito and Remy[2] introduced polyharmonic local sine transform (PHLST). The essential difference between PHLST and DCT is as follows.

For PHLST, the function defined on a square is decomposed into two parts. The first part is the polyharmonic component which is a solution of the polyharmonic equation $\Delta^m u = 0$ given the boundary condition. The second part is the residual that vanishes on the boundary. Afterwards we extend the residual by "odd" reflection at the boundary. For $m = 1$, the polyharmonic equation is reduced to Laplace's equation with Dirichlet boundary condition. For the two-dimensional case, Averbuch, Israeli, and Vozovoi[3] derived the fast and accurate solution of Laplace's equation with Dirichlet boundary condition. Using this method, we obtain the polyharmonic component. This odd extension of the residual is a continuously differentiable function across the square boundaries. We expand it into the Fourier "sine" series. Now the rate of decay of the Fourier coefficients is $O(\|\boldsymbol{\ell}\|^{-3})$, where $\boldsymbol{\ell} = (\ell_1, \ell_2)$ is the coefficient index and $\|\boldsymbol{\ell}\| = \sqrt{\ell_1^2 + \ell_2^2}$.

To improve this rate of decay, we need to take the large integer m. For $m = 2$, the rate of decay of the Fourier coefficients of the residual is $O(\|\boldsymbol{\ell}\|^{-5})$. However one needs to solve $\Delta^2 u = 0$ with second order normal derivative at boundaries, which is very difficult to estimate in applications.

In order to compensate this problem of PHLST, Yamatani and Saito presented a new image compression method: the polyharmonic local cosine transforms (PHLCT)[1] that decomposes an image into a polyharmonic component and a residual. The polyharmonic component is a solution of Poisson's equation with the Neumann boundary condition. Subsequently the polyharmonic component is subtracted from the original function to obtain the residual component. We extend the residual component by "even" reflection at the boundary. This even extension of the residual is a twice continuously differentiable function across the square boundaries. We expand it into the Fourier cosine series, and the decay rate of the Fourier cosine

Further author information: (Send correspondence to Z.Z.)
Z.Z.: E-mail: zzh@ucdavis.edu
N.S.: E-mail: saito@math.ucdavis.edu, URL: http://www.math.ucdavis.edu/˜saito

coefficients of the residual is $O(\|\boldsymbol{\ell}\|^{-4})$. Thus, it is clear that PHLCT algorithm can compress data more efficiently than the PHLST and DCT algorithms.

In the field of data analysis, one often needs to deal with high-dimensional data, e.g., 3D marine seismic data, medical tomographic data, etc. In the present paper, we will generalize PHLCT to the high-dimensional case and apply it to compress such high-dimensional data. We will first investigate the computational issue of the polyharmonic component of the high-dimensional data f defined in $\mathbb{R}^d$ with $d \geq 3$. For this purpose, we must give the series solution of the high-dimensional version of Poisson's equation with the Neumann boundary condition. This is an important and difficult problem, and it has its own interest. We give a satisfying answer. Based on the partition of boundaries of the cube, the solution given by us has a simple and clear representation and it is convenient to apply it to high-dimensional data compression problems. When we use PHLCT to compress a d-dimensional data, in order to reduce the number of the coefficients of PHLCT, we use not only d-dimensional PHLCT decomposition, but also $d-1, d-2, \ldots, 1$ dimensional PHLCT decomposition. The number of PHLCT coefficients obtained by us is equal to the number of the original sample points. These coefficients decay very fast and can recover the data exactly. Therefore, we can approximate the original data quite efficiently by truncating the coefficients whose magnitude is small.

At the end of this paper, we use PHLCT and DCT to approximate 3D Gaussian data and real 3D seismic data. From the results of these numerical experiments, we see that PHLCT can compress high-dimensional data more efficiently than DCT.

2. THE DCT ALGORITHM AND THE PHLCT ALGORITHM

In this section, we will review two important algorithms in image compression. First we introduce *Discrete Cosine Transform* (DCT). DCT is very important in history.[5] In theory, DCT can approximate the Karhunen-Loève Transform (KLT) if the signal is the 1st order Markov random process.[7] In application, DCT can efficiently compress data.[8] The well-known image compression standard JPEG is just based on it.[6] To state the idea of DCT, we need the following notations.

Let $\zeta_j = \frac{0.5+j}{N}, \; j = 0, 1, .., N-1$ and N_d be the set of all the integer lattice points in the cube $[0, N-1]^d$.

Let $\Omega = [0,1]^d$ and f be a function defined on Ω. Let $\boldsymbol{k} = (k_1, \ldots, k_d) \in N_d$ and $\boldsymbol{x_k} = (\zeta_{k_1}, \ldots, \zeta_{k_d}) \in \Omega$. Then the DCT coefficients of f is defined as

$$b_{\boldsymbol{\ell}} = C_{\boldsymbol{\ell}}\, N^{-d/2} \sum_{\boldsymbol{k} \in N_d} f(\boldsymbol{x_k}) \prod_{i=1}^{d} \cos(\pi \ell_i \, \zeta_{k_i}) \qquad \boldsymbol{\ell} = (\ell_1, \ldots, \ell_d) \in N_d, \tag{1}$$

where $C_{\boldsymbol{\ell}} = 2^{\Sigma_{i=1}^d \alpha(\ell_i)/2}$ and $\alpha(i) = 1\,(i \neq 0),\; \alpha(0) = 0$. The decay rate of the DCT coefficients is $O(\,\|\boldsymbol{\ell}\|^{-2})$, $\|\boldsymbol{\ell}\| = \left(\ell_1^2 + \cdots + \ell_d^2\right)^{\frac{1}{2}}$.

In 2006, Yamatani and Saito[1] presented the polyharmonic local cosine transforms (PHLCT) as follows. Let f be a function defined on $\Omega = [0,1]^d$ and q be the normal derivative $\frac{\partial u}{\partial \boldsymbol{n}}$ of f on $\partial\Omega$. The *Polyharmonic Local Cosine Transform* (PHLCT) decomposes f into two components as $f = u + v$, where u satisfies Poisson's equation with the Neumann boundary condition

$$\varDelta u = I(q), \quad \boldsymbol{x} \in \Omega, \qquad \frac{\partial u}{\partial \boldsymbol{n}} = q, \quad \boldsymbol{x} \in \partial\Omega, \tag{2}$$

and $I(q) = \int\limits_{\partial\Omega} q(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}$. We call u the polyharmonic component and v the residual.

For $d = 2$, Yamatani and Saito[1] gave the solution of the equation (2). Because the residual has the zero normal derivatives on the boundary, the even extension of f has desired C^2 smoothness if there is no intrinsic singularity in the original data f. Consequently, the DCT coefficients of the residual will decay as fast as $O(\|\boldsymbol{\ell}\|^{-4})$. From this, we see that the PHLCT algorithm can compress the two-dimensional data more efficiently than the DCT algorithm both in theory and in application.

3. D-DIMENSIONAL POLYHARMONIC LOCAL COSINE TRANSFORM

Now we generalize polyharmonic local cosine transform into the high-dimensional case. The key is to solve Poisson's equation with the Neumann boundary condition in (2). For the two-dimensional case, Averbuch, Israeli, and Vozovoi[3] derived the solution of Laplace's equation with the Dirichlet or the Neumann boundary conditions. Later on, Yamatani and Saito[1] modified it and apply it to image compression. For the three-dimensional case, Braverman, Israeli, Averbuch, and Vozovoi[4] derived the solution of Poisson's equation with the Dirichlet boundary condition. Below we derive the solution of Poisson's equation with the Neumann boundary condition for the three-dimensional and the higher-dimensional cases.

Let f be defined on $\Omega = [0,1]^d$ and q be the normal derivative of f on $\partial\Omega$, i.e., $\frac{\partial f}{\partial \boldsymbol{n}} = q$. Now we give the solution of the high-dimensional Poisson equation (2) with the Neumann boundary condition.

Notation 3.1. We define the following notations.

(i) Let N_d be stated as above. For each $i = 1, \dots, d$, denote the subsets of N_d

$$\boldsymbol{H}_i^0 : = \{ \ \boldsymbol{k} = (k_1, \dots, k_d) \quad | \quad k_i = 0, \ 0 \le k_j \le N-1, \ \forall j \ne i \ \},$$

$$\boldsymbol{H}_i^1 : = \{ \ \boldsymbol{k} = (k_1, \dots, k_d) \quad | \quad k_i = N-1, \ 0 \le k_j \le N-1, \ \forall j \ne i \ \}.$$

(ii) Let $\Omega = [0,1]^d$. For each $i = 1, \dots, d$, denote the faces of the cube Ω

$$\boldsymbol{R}^{(i,0)} : = \{ \ \boldsymbol{x} = (x_1, \dots, x_d) \in \Omega \quad | \quad x_i = 0 \ \},$$

$$\boldsymbol{R}^{(i,1)} : = \{ \ \boldsymbol{x} = (x_1, \dots, x_d) \in \Omega \quad | \quad x_i = 1 \ \}.$$

From this, we know that

$$\partial N_d := N_d \bigcap \partial\big([0, N-1]^d\big) = \bigcup_{i=1}^{d} \left(\boldsymbol{H}_i^0 \bigcup \boldsymbol{H}_i^1 \right), \qquad \partial\Omega = \bigcup_{i=1}^{d} \left(\boldsymbol{R}^{(i,0)} \bigcup \boldsymbol{R}^{(i,1)} \right). \tag{3}$$

We assume that the discretized data $\{ \ q(\boldsymbol{x_k}) \ \}_{\boldsymbol{k} \in \partial N_d}$ of $q(\boldsymbol{x})$ on boundary $\partial\Omega$ are available, where

$$\boldsymbol{x_k} = (\zeta_{k_1}, \dots, \zeta_{k_d}) \quad \text{and} \quad \zeta_{k_\nu} = \frac{0.5 + k_\nu}{N}, \quad \nu = 1, 2, \dots, d.$$

Denote $q_{\boldsymbol{k}}^{(i,\tau)} : = q(\boldsymbol{x_k}), \ \ \boldsymbol{k} \in \boldsymbol{H}_i^\tau$, which is a sample of $q(\boldsymbol{x})$ at $\boldsymbol{x_k} \in \boldsymbol{R}^{(i,\tau)}$.

Let $B_{\boldsymbol{k}}^{(i,\tau)}, \ \ \boldsymbol{k} \in \boldsymbol{H}_i^\tau$ be the DCT coefficients of $q_{\boldsymbol{k}}^{(i,\tau)}, \ \ \boldsymbol{k} \in \boldsymbol{H}_i^\tau$. Define

$$q^{(i,\tau)}(\boldsymbol{x}) = \left(\frac{1}{N}\right)^{\frac{d-1}{2}} \sum_{\boldsymbol{k} \in \boldsymbol{H}_i^\tau} A_{\boldsymbol{k}} B_{\boldsymbol{k}}^{(i,\tau)} \prod_{\nu \ne i} \cos(\pi \, k_\nu \, x_\nu), \quad \boldsymbol{x} = (x_1, \dots, x_d), \tag{4}$$

where

$$A_{\boldsymbol{k}} = 2^{\sum\limits_{\substack{1 \le \ell \le d \\ \ell \ne i}} \alpha(k_\ell)/2}.$$

and $\alpha(0) = 0, \ \alpha(i) = 1, i \ne 0$.

By inverse DCT, we know that $q^{(i,\tau)}(\boldsymbol{x})$ is a trigonometric polynomial approximating $q(\boldsymbol{x})$ on the face $\boldsymbol{R}^{(i,\tau)}$. Hence

$$q(\boldsymbol{x}) \approx q^{(i,\tau)}(\boldsymbol{x}), \qquad \boldsymbol{x} \in \boldsymbol{R}^{(i,\tau)}, \quad i = 1, \dots, d, \ \ \tau = 0, \, 1. \tag{5}$$

Theorem 3.2. The approximate solution of Poisson's equation with the Neumann boundary condition

$$\varDelta u = I(q), \quad \boldsymbol{x} \in \Omega, \qquad \frac{\partial u}{\partial \boldsymbol{n}} = q, \quad \boldsymbol{x} \in \partial\Omega \qquad \left(I(q) = \int\limits_{\partial\Omega} q(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \right)$$

is written as

$$u(\boldsymbol{x}) = \sum_{i=1}^{d} (u^{(i,0)}(\boldsymbol{x}) + u^{(i,1)}(\boldsymbol{x})),$$

where

$$u^{(i,0)}(\boldsymbol{x}) := \left(\frac{1}{N}\right)^{\frac{d-1}{2}} \left\{ A_{\boldsymbol{0}} B_{\boldsymbol{0}}^{(i,0)} \frac{(x_i-1)^2}{2} + \sum_{\substack{\boldsymbol{k}\in \boldsymbol{H}_i^0 \\ \boldsymbol{k}\neq \boldsymbol{0}}} W_1 W_3 \right\} + C, \tag{6}$$

$$u^{(i,1)}(\boldsymbol{x}) := \left(\frac{1}{N}\right)^{\frac{d-1}{2}} \left\{ A_{\boldsymbol{0}} B_{\boldsymbol{0}}^{(i,1)} \frac{x_i^2}{2} + \sum_{\substack{\boldsymbol{k}\in \boldsymbol{H}_i^1 \\ \boldsymbol{k}\neq 0}} W_3 W_2 \right\} + C, \qquad \boldsymbol{x} = (x_1, \ldots, x_d),$$

here C are different constants and

$$\begin{aligned} W_1 &: = W_{1,\boldsymbol{k}}^i(\boldsymbol{x}) : = \alpha_{\boldsymbol{k}} B_{\boldsymbol{k}}^{(i,0)} \cosh(\pi \eta_{\boldsymbol{k}} (x_i - 1)), \\ W_2 &: = W_{2,\boldsymbol{k}}^i(\boldsymbol{x}) : = \alpha_{\boldsymbol{k}} B_{\boldsymbol{k}}^{(i,1)} \cosh(\pi \eta_{\boldsymbol{k}} x_i), \qquad \alpha_{\boldsymbol{k}} = \frac{A_{\boldsymbol{k}}}{\pi \eta_{\boldsymbol{k}} \sinh(\pi \eta_{\boldsymbol{k}})}, \quad \eta_{\boldsymbol{k}} = \left(\sum_{\ell \neq i} k_\ell^2 \right)^{1/2}, \\ W_3 &: = W_{3,\boldsymbol{k}}^i(\boldsymbol{x}) : = \prod_{\substack{1\leq \ell \leq d \\ \ell \neq i}} \cos(\pi\, k_\ell\, x_\ell). \end{aligned} \tag{7}$$

Proof. By (3), (5), and (6), we only need to prove that for each $i = 1, \ldots, d,\ \tau = 0, 1,$

$$\varDelta u^{(i,\tau)} = \int\limits_{\boldsymbol{R}^{(i,\tau)}} q^{(i,\tau)}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}, \quad \boldsymbol{x} \in \Omega \tag{8}$$

and

$$\frac{\partial u^{(i,\tau)}}{\partial \boldsymbol{n}} = q^{(i,\tau)}, \quad \boldsymbol{x} \in \boldsymbol{R}^{(i,\tau)}, \qquad \frac{\partial u^{(i,\tau)}}{\partial \boldsymbol{n}} = 0, \quad \boldsymbol{x} \in \partial\Omega \setminus \boldsymbol{R}^{(i,\tau)}. \tag{9}$$

By the similarity, we only prove the case $\tau = 0$. First we prove (8) for $\tau = 0$.

Since the Laplace operator $\varDelta$ is linear, $\varDelta C = 0$, and $\varDelta\left(\frac{(x_i-1)^2}{2}\right) = 1$, we conclude by (6) that for each $i = 1, \ldots, d,$

$$\varDelta u^{(i,0)}(\boldsymbol{x}) = \left(\frac{1}{N}\right)^{\frac{d-1}{2}} \left(A_{\boldsymbol{0}} B_{\boldsymbol{0}}^{(i,0)} + \sum_{\substack{\boldsymbol{k}\in \boldsymbol{H}_i^0 \\ \boldsymbol{k}\neq 0}} \varDelta(W_1 W_3) \right). \tag{10}$$

From the definitions of the Laplace operator and the gradient operator,

$$\varDelta W := \left(\frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + \cdots + \frac{\partial^2}{\partial x_d^2} \right) W, \qquad \nabla W := \left(\frac{\partial}{\partial x_1},\ \frac{\partial}{\partial x_2},\ \ldots,\ \frac{\partial}{\partial x_d} \right) W,$$

we can check directly the following formula

$$\varDelta(W_1 W_3) = (\varDelta W_1) W_3 + (\varDelta W_3) W_1 + 2 \nabla W_1 \cdot \nabla W_3. \tag{11}$$

This formula gives a great convenience in our calculation. By a direct calculation using (7), we have

$$\frac{\partial W_1}{\partial x_i} = \alpha_{\boldsymbol{k}} B_{\boldsymbol{k}}^{(i,0)} \pi \eta_{\boldsymbol{k}} \sinh\left(\pi \eta_{\boldsymbol{k}} (x_i - 1)\right), \qquad \frac{\partial^2 W_1}{\partial x_i^2} = \alpha_{\boldsymbol{k}} B_{\boldsymbol{k}}^{(i,0)} \pi^2 \eta_{\boldsymbol{k}}^2 \cosh\left(\pi \eta_{\boldsymbol{k}} (x_i - 1)\right), \tag{12}$$

$$\frac{\partial W_1}{\partial x_\nu} = \frac{\partial^2 W_1}{\partial x_\nu^2} = 0, \qquad \nu \neq i \tag{13}$$

and

$$\frac{\partial W_3}{\partial x_i} = 0, \qquad \frac{\partial^2 W_3}{\partial x_\nu^2} = -\left(\prod_{\ell\neq i}\cos(\pi k_\ell x_\ell)\right)\pi^2 k_\nu^2, \quad \nu \neq i.$$

Hence by (11), $\mathit{\Delta}(W_1 W_3) = 0$. Furthermore, by (10), we have

$$\mathit{\Delta} u^{(i,0)}(\boldsymbol{x}) = \left(\frac{1}{N}\right)^{\frac{d-1}{2}} A_{\boldsymbol{0}} B_{\boldsymbol{0}}^{(i,0)}. \tag{14}$$

On the other hand, by Notation 3.1 (ii), we know that $\boldsymbol{x} = (x_1, \dots, x_d) \in \boldsymbol{R}^{(i,0)}$ if and only if $0 \leq x_\nu \leq 1$ $(\nu \neq i)$ and $x_i = 0$. From this and (4), we have

$$\int\limits_{\boldsymbol{R}^{(i,0)}} q^{(i,0)}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \left(\frac{1}{N}\right)^{\frac{d-1}{2}} \sum_{\boldsymbol{k}\in \boldsymbol{H}_i^0} A_{\boldsymbol{k}} B_{\boldsymbol{k}}^{(i,0)} \left(\prod_{\nu\neq i}\int_0^1 \cos(\pi\, k_\nu\, x_\nu)\,\mathrm{d}x_\nu\right), \quad \boldsymbol{x} = (x_1, \dots, x_d),\ \ \boldsymbol{k} = (k_1, \dots, k_d).$$

If $\boldsymbol{k} \in \boldsymbol{H}_i^0$ and $\boldsymbol{k} \neq 0$, then there exists a $\nu \neq i$ such that $k_\nu \neq 0$, so $\int\limits_0^1 \cos(\pi\, k_\nu\, x_\nu)\mathrm{d}x_\nu = 0$. From this, we can conclude that

$$\int\limits_{\boldsymbol{R}^{(i,0)}} q^{(i,0)}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \left(\frac{1}{N}\right)^{\frac{d-1}{2}} A_{\boldsymbol{0}} B_{\boldsymbol{0}}^{(i,0)}.$$

Again by (14), we obtain (8) for $\tau = 0$.

Below we prove that the formula (9) holds for $\tau = 0$. Since the gradient operator is linear, by (6) and $\nabla\left(\frac{(x_i-1)^2}{2}\right) = (x_i - 1)\boldsymbol{e}_i$, we have

$$\nabla u^{(i,0)}(\boldsymbol{x}) = \left(\frac{1}{N}\right)^{\frac{d-1}{2}} \left(A_{\boldsymbol{0}} B_{\boldsymbol{0}}^{(i,0)}(x_i - 1)\boldsymbol{e}_i + \sum_{\substack{\boldsymbol{k}\in \boldsymbol{H}_i^0 \\ \boldsymbol{k}\neq 0}} \nabla(W_1 W_3)\right), \tag{15}$$

where $\boldsymbol{e}_i$ is an unit vector in $\mathbb{R}^d$ whose ith component is 1. Let $\boldsymbol{n}$ be the outward normal vector of $\partial\Omega$. Then, by (15), we have

$$\frac{\partial u}{\partial \boldsymbol{n}} = \boldsymbol{n} \cdot \nabla u^{(i,0)} = \left(\frac{1}{N}\right)^{\frac{d-1}{2}} \left(A_{\boldsymbol{0}} B_{\boldsymbol{0}}^{(i,0)}(\boldsymbol{n},\ (x_i - 1)\boldsymbol{e}_i) + \sum_{\substack{\boldsymbol{k}\in \boldsymbol{H}_i^0 \\ \boldsymbol{k}\neq 0}} \boldsymbol{n} \cdot \nabla(W_1 W_3)\right). \tag{16}$$

By Notation 3.1 (ii), we know that

$$\text{if } \ \boldsymbol{x} \in \partial\Omega \setminus \boldsymbol{R}^{(i,0)}, \qquad \text{then } \boldsymbol{n} \perp \boldsymbol{e}_i \ \text{ or } \ x_i = 1;$$
$$\text{if } \ \boldsymbol{x} \in \boldsymbol{R}^{(i,0)}, \qquad \text{then } \ \boldsymbol{n} = -\boldsymbol{e}_i \text{ and } \ x_i = 0.$$

So we have

$$\boldsymbol{n} \cdot (x_i - 1)\boldsymbol{e}_i = 0, \quad \boldsymbol{x} \in \partial\Omega \setminus \boldsymbol{R}^{(i,0)} \qquad \text{and} \qquad \boldsymbol{n} \cdot (x_i - 1)\boldsymbol{e}_i = -(x_i - 1) = 1, \quad \boldsymbol{x} \in \boldsymbol{R}^{(i,0)}. \tag{17}$$

Using the formula

$$\nabla(W_1 W_3) = (\nabla W_1) W_3 + (\nabla W_3) W_1, \tag{18}$$

together with (12) and (13), we have

$$\nabla W_1 = \alpha_{\boldsymbol{k}} B_{\boldsymbol{k}}^{(i,0)} \pi \eta_{\boldsymbol{k}} \sinh(\pi\, \eta_{\boldsymbol{k}}(x_i - 1))\, \boldsymbol{e}_i. \tag{19}$$

For $\boldsymbol{x} \in \boldsymbol{R}^{(i,0)},\ x_i = 0$, by (7), we have

$$\nabla W_1 = -\alpha_{\boldsymbol{k}} B_{\boldsymbol{k}}^{(i,0)} \pi \eta_{\boldsymbol{k}} \sinh(\pi \eta_{\boldsymbol{k}}) \boldsymbol{e}_i = -A_{\boldsymbol{k}} B_{\boldsymbol{k}}^{(i,0)} \boldsymbol{e}_i.$$

Furthermore, by $(\boldsymbol{n},\, \boldsymbol{e}_i) = -1$ and (7), we obtain that

$$\boldsymbol{n} \cdot ((\nabla W_1) W_3) = A_{\boldsymbol{k}} B_{\boldsymbol{k}}^{(i,0)} \prod_{\ell \neq i} \cos(\pi k_\ell x_\ell). \tag{20}$$

Since $\frac{\partial W_3}{\partial x_i} = 0$ and $\boldsymbol{n} = -\boldsymbol{e}_i$, we have $\boldsymbol{n} \cdot \nabla W_3 = 0$. So

$$\boldsymbol{n} \cdot ((\nabla W_3) W_1) = 0. \tag{21}$$

Hence on $\boldsymbol{R}^{(i,0)}$, by (18)–(20),

$$\boldsymbol{n} \cdot \nabla (W_1 W_3) = A_{\boldsymbol{k}} B_{\boldsymbol{k}}^{(i,0)} \prod_{\ell \neq i} \cos(\pi k_\ell x_\ell).$$

From this and (16)-(17), we obtain that

$$\frac{\partial u}{\partial \boldsymbol{n}} = \left(\frac{1}{N} \right)^{\frac{d-1}{2}} \left(A_{\boldsymbol{0}} B_{\boldsymbol{0}}^{(i,0)} + \sum_{\substack{\boldsymbol{k} \in \boldsymbol{H}_i^0 \\ \boldsymbol{k} \neq 0}} A_{\boldsymbol{k}} B_{\boldsymbol{k}}^{(i,0)} \prod_{\ell \neq i} \cos(\pi k_\ell x_\ell) \right), \quad \boldsymbol{x} \in \boldsymbol{R}^{(i,0)}.$$

For $\boldsymbol{k} = 0$, we know that each $k_\nu = 0$, so $\prod\limits_{\nu \neq i} \cos(\pi k_\nu x_\nu) = 1$. From this and (4), we obtain that

$$\frac{\partial u}{\partial \boldsymbol{n}} = \left(\frac{1}{N} \right)^{\frac{d-1}{2}} \sum_{\boldsymbol{k} \in \boldsymbol{H}_i^0} A_{\boldsymbol{k}} B_{\boldsymbol{k}}^{(i,0)} \left(\prod_{\ell \neq i} \cos(\pi k_\ell x_\ell) \right) = q^{(i,0)}(\boldsymbol{x}), \quad \boldsymbol{x} \in \boldsymbol{R}^{(i,0)}. \tag{22}$$

For $\boldsymbol{x} \in \boldsymbol{R}^{(i,1)},\ x_i = 1$, by (19), we have

$$\nabla W_1 = 0,\ \boldsymbol{x} \in \boldsymbol{R}^{(i,1)}.$$

So $\boldsymbol{n} \cdot ((\nabla W_1) W_3) = 0$. By (7) and (18), we have

$$\boldsymbol{n} \cdot \nabla (W_1 W_3) = \boldsymbol{n} \cdot ((\nabla W_1) W_3) + \boldsymbol{n} \cdot ((\nabla W_3) W_1) = 0, \quad \boldsymbol{x} \in \boldsymbol{R}^{(i,1)}.$$

Again by (16) and (17), we obtain that

$$\frac{\partial u}{\partial \boldsymbol{n}} = \boldsymbol{n} \cdot \nabla u^{(i,0)} = 0, \quad \boldsymbol{x} \in \boldsymbol{R}^{(i,1)}. \tag{23}$$

For $\boldsymbol{x} \in \partial\Omega$ and $\boldsymbol{x} \notin (\boldsymbol{R}^{(i,0)} \bigcup \boldsymbol{R}^{(i,1)})$, we have $\boldsymbol{n} \perp \boldsymbol{e}_i$. Furthermore, by (19),

$$\boldsymbol{n} \cdot ((\nabla W_1) W_3) = 0. \tag{24}$$

For $\boldsymbol{x} = (x_1, \ldots, x_d) \in \partial\Omega \setminus (\boldsymbol{R}^{(i,0)} \bigcup \boldsymbol{R}^{(i,1)})$, there exists a $\nu \neq i$ such that $x_\nu = 1$ or $x_\nu = 0$, and the normal vector of $\partial\Omega$ at $\boldsymbol{x}$ is $\boldsymbol{e}_\nu$ or $-\boldsymbol{e}_\nu$, so $\boldsymbol{n} = \boldsymbol{e}_\nu$ or $-\boldsymbol{e}_\nu$. By (7), we have

$$\frac{\partial W_3}{\partial x_\nu} = - \left(\prod_{\substack{1 \leq \ell \leq d \\ \ell \neq \nu,\ i}} \cos(\pi k_l x_l) \right) \pi k_\nu \sin(\pi k_\nu x_\nu),$$

so $\boldsymbol{n} \cdot \nabla W_3 = \pm \frac{\partial W_3}{\partial x_\nu} = 0.$ Furthermore, by (18) and (24), we obtain $\boldsymbol{n} \cdot \nabla (W_1 W_3) = 0.$ Again, by (16) and (17), we have

$$\frac{\partial u}{\partial \boldsymbol{n}} = 0, \qquad \boldsymbol{x} \in \partial\Omega \setminus (\, \boldsymbol{R}^{(i,0)} \bigcup \boldsymbol{R}^{(i,1)}\,). \tag{25}$$

Combining (22), (23) with (25), we conclude that (9) holds for $\tau = 0$. Theorem 3.2 is proved.

Let f be defined on $\Omega = [0,1]^d$. Using Theorem 3.2, we can decompose f into the polyharmonic component u and the residual v, i.e., $f(\boldsymbol{x}) = u(\boldsymbol{x}) + v(\boldsymbol{x}),\ \boldsymbol{x} \in \Omega$.

4. HIGH-DIMENSIONAL DATA COMPRESSION ALGORITHM

In this section, we will present our new algorithm of high-dimensional data compression. In order to reduce the number of PHLCT coefficients, we need to use not only d-dimensional PHLCT decomposition, but also $d-1,\ d-2,\dots,1$ dimensional PHLCT decomposition. First we introduce the following notations.

Let $N_d = [0,\ N-1]^d \cap \mathbb{Z}_d$ be stated in Section 2. We divide N_d into $d+1$ sets.

Notation 4.1. $\boldsymbol{D} :=\{1,2,3,\dots..d\}$.

Notation 4.2. Let $\boldsymbol{e}$ be a subset of the set $\boldsymbol{D}$. Denote the number of the elements in $\boldsymbol{e}$ by $|\boldsymbol{e}|$.

Notation 4.3. Let $\boldsymbol{\ell} = (\ell_1,\dots\dots,\ell_d) \in N_d$. If $\ell_i = 0,\ i\in \boldsymbol{e}$ and $\ell_i \neq 0,\ i \notin \boldsymbol{e}$, then we say $\boldsymbol{\ell} \in \boldsymbol{D}_{\boldsymbol{e}}$.

Notation 4.4. $\boldsymbol{D}_r := \{\, \boldsymbol{D}_{\boldsymbol{e}},\ \ |\boldsymbol{e}| = r\,\}$, where $\boldsymbol{D}_0 = \{\, \boldsymbol{\ell} \in N_d,\ \text{each}\ \ \ell_i \neq 0\,\}$.

Proposition 4.5. $N_d = \bigcup\limits_{r=0}^{d} \boldsymbol{D}_r$.

Let f be defined on $[0,1]^d$. Assume that the discretized version of f sampled at $\boldsymbol{x_k},\ \ \boldsymbol{k} \in N_d$ are given, where $\boldsymbol{x_k} = \{\zeta_{k_1},\dots,\zeta_{k_d}\}$ and $\zeta_{k_i} = \frac{0.5+k_i}{N},\ \boldsymbol{k} = (k_1,\dots,k_d)$.

Proposition 4.6. Let f be defined on $\Omega=[0,1]^d$, and let $\boldsymbol{\ell} \in \boldsymbol{D}_{\boldsymbol{e}},\ \ |\boldsymbol{e}| = r$ and $\boldsymbol{D}\setminus \boldsymbol{e} = (i_1,\dots,i_{d-r})$. Then the DCT coefficient $b_{\boldsymbol{\ell}}$ $(\boldsymbol{\ell} \in \boldsymbol{D}_{\boldsymbol{e}})$ of f is equal to the DCT coefficient of the $(d-r)-$dimensional function $h(\eta_1,\dots,\eta_{d-r})$ defined as

$$h(\zeta_{k_{i_1}},\dots,\zeta_{k_{i_{d-r}}}) = N^{-\frac{r}{2}} \sum_{i\in e}\sum_{k_i} f(\zeta_{k_1},\ \dots,\ \zeta_{k_d}). \tag{26}$$

Proof. For $\boldsymbol{\ell} = (\ell_1,\dots,\ell_d) \in \boldsymbol{D}_{\boldsymbol{e}}$, by Notation 4.3, we have $\ell_i = 0,\ \ i\in\boldsymbol{e}$. By (1), we have

$$b_{\boldsymbol{\ell}} = C_{\boldsymbol{\ell}} N^{-\frac{d}{2}} \sum_{\boldsymbol{k}\in N_d} f(\boldsymbol{x_k}) \prod_{i\notin \boldsymbol{e}} \cos(\pi\,\ell_i\,\zeta_{k_i}), \quad \boldsymbol{k} = (k_1,\dots,k_d),\ \ \boldsymbol{x_k} = (\zeta_{k_1},\dots,\zeta_{k_d}) \tag{27}$$

where $C_{\boldsymbol{\ell}} = 2^{\sum_{i=1}^{d}\alpha(\ell_i)/2}$. In (27), the product $\prod\limits_{i\notin\boldsymbol{e}} \cos(\pi\,\ell_i\,\zeta_{k_i})$ does not depend on $\zeta_{k_i},\ \ i\in\boldsymbol{e}$. By Notation 4.3, we have

$$\begin{aligned} b_{\boldsymbol{\ell}} &= C_{\boldsymbol{\ell}} N^{-\frac{d}{2}} \sum_{k_i, i\notin\boldsymbol{e}}\ \sum_{k_i, i\in\boldsymbol{e}} \left(f(\zeta_{k_1},\dots,\zeta_{k_d}) \prod_{i\notin\boldsymbol{e}} \cos(\pi\,\ell_i\,\zeta_{k_i}) \right) \\ &= C_{\boldsymbol{\ell}} N^{-\frac{d-r}{2}} \sum_{k_i, i\notin\boldsymbol{e}} \left(\left(\ N^{-\frac{r}{2}} \sum_{k_i, i\in\boldsymbol{e}} f(\zeta_{k_1},\dots,\zeta_{k_d})\ \right) \prod_{i\notin\boldsymbol{e}} \cos(\pi\,\ell_i\,\zeta_{k_i}) \right). \end{aligned}$$

From this and (26), we obtain that

$$b_{\boldsymbol{\ell}} = C_{\boldsymbol{\ell}} N^{-\frac{d-r}{2}} \sum_{k_i, i\notin\boldsymbol{e}} \left(h(\zeta_{k_{i_1}},\dots,\zeta_{k_{i_{d-r}}}) \prod_{i\notin\boldsymbol{e}} \cos(\pi\,\ell_i\,\zeta_{k_i}) \right), \tag{28}$$

where h is stated in (26). Since $\ell_i = 0,\ \ i\in\mathbf{e}$, we have $\sum\limits_{i=1}^{d}\alpha(\ell_i)/2 = \sum\limits_{i\notin\boldsymbol{e}}\alpha(\ell_i)/2$. So

$$C_{\boldsymbol{\ell}} = 2^{\sum_{i=1}^{d}\alpha(\ell_i)/2} = 2^{\sum_{i\notin\boldsymbol{e}}\alpha(\ell_i)/2}.$$

From this and (28), noticing that $|\boldsymbol{e}| = r$, we know that $b_{\boldsymbol{\ell}}$ is the DCT coefficient of the $(d-r)-$dimensional function $h(\eta_1,\dots,\eta_{d-r})$. Proposition 4.6 is proved.

Now, for the function f defined on $\Omega=[0,1]^d$, we present the following compression algorithm to compute the PHLCT coefficients in processes of repeated PHLCT-decompositions.

Step 0. Compute DCT coefficients $b_{\boldsymbol{\ell}}$ $,\boldsymbol{\ell} \in N_d$ of f.

Step 1. Consider DCT coefficients $b_{\boldsymbol{\ell}}$ $(\boldsymbol{\ell} \in \boldsymbol{D}_0)$ of f. We decompose the d-dimensional function as $f = u^0 + v^0$, where u^0 is the solution of the Poisson equation in Theorem 3.2 and v^0 is the residual. Compute the DCT coefficient $q_{\boldsymbol{\ell}}$ of u^0, we will obtain the PHLCT coefficient $p_{\boldsymbol{\ell}} = b_{\boldsymbol{\ell}} - q_{\boldsymbol{\ell}},\ \boldsymbol{\ell} \in D_0$.

. . . .

Step $(r+1)$**.** Consider DCT coefficients $b_{\boldsymbol{\ell}}$ $(\boldsymbol{\ell} \in D_r)$ of f. By Notation 4.4, we have $\boldsymbol{D_r} = \{\, \boldsymbol{D_e},\ \ |\boldsymbol{e}| = r\,\}$. For $\boldsymbol{l} \in \boldsymbol{D_e}$ and $\boldsymbol{D} \setminus \boldsymbol{e} = (i_1, \ldots, i_{d-r})$, by Proposition 4.6, we see that $b_{\boldsymbol{\ell}}$ is the DCT coefficient of the $(d-r)-$dimensional function h which is stated in (26). Now we use $(d-r)-$dimensional PHLCT to decompose the $(d-r)-$dimensional function h into $h = u^r + v^r$. Compute the DCT coefficient $q_{\boldsymbol{\ell}}$ of u^r, we will obtain the PHLCT coefficient $p_{\boldsymbol{\ell}} = b_{\boldsymbol{\ell}} - q_{\boldsymbol{\ell}},\ \boldsymbol{\ell} \in \boldsymbol{D_e}$.

. . . .

Step d**.** Consider DCT coefficients $b_{\boldsymbol{\ell}}$ $(\boldsymbol{\ell} \in \boldsymbol{D}_{d-1})$ of f. Similar to the argument of Step $(r+1)$, applying the one-dimensional PHLCT decomposition, we can compute PHLCT coefficients $p_{\boldsymbol{\ell}}$, where $\boldsymbol{\ell} \in \boldsymbol{D}_{d-1}$.

Step $(d+1)$**.** Consider the DCT coefficient $b_{\mathbf{0}}$ of f. The PHLCT coefficient $p_{\mathbf{0}} = b_{\mathbf{0}}$.

Remark 4.7. Using these PHLCT coefficients, we can also reconstruct the data perfectly. Since the reconstruction algorithm is simply the inverse of the above compression algorithm, we will not state it here.

5. NUMERICAL EXPERIMENTS

In this section, we demonstrate the effectiveness of our algorithm by numerical experiments. We split high-dimensional data into small blocks and apply high-dimensional PHLCT algorithm on every block to approximate the data. Our data approximation strategy is the following: 1) retain all "DC" components; 2) select a certain number of the largest PHLCT coefficients in energy among all the PHLCT coefficients of all the blocks; and 3) reconstruct the data from these retained coefficients. The quality of approximation is measured by PSNR (peak signal-to-noise ratio).

In order to demonstrate the effectiveness of our algorithm, first we use PHLCT and DCT to approximate 3D Gaussian data. The size of 3D Gaussian data is $128 \times 128 \times 128$. We split it into a set of small blocks each of which has $8 \times 8 \times 8$ samples. Table 1 shows PSNR values of the 3D Gaussian data approximated by PHLCT and DCT. The PSNR values of PHLCT are 25 dB higher than those of DCT. From this, we can see the clear superiority of 3D PHLCT over 3D DCT for such a smooth dataset.

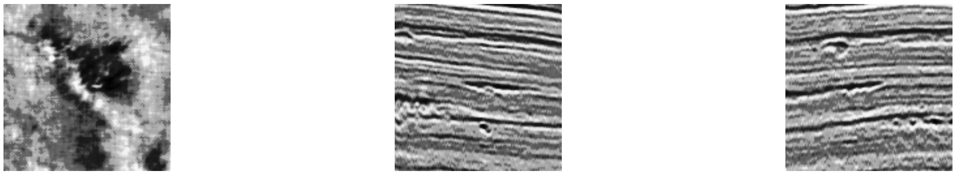
Table 1: PSNR values (dB) of 3D Gaussian data approximated by 3D PHLCT and 3D DCT

Ratio of retained coefficients	1.0%	1.5%	2.0%	2.5%
PHLCT	86.5990	93.3359	98.4086	102.5149
DCT	61.3616	67.7792	72.4882	76.4468

Now, we examine the approximation performance of these transforms using real 3D real seismic data. The size of 3D seismic data is also $128 \times 128 \times 128$. Table 2 shows PSNR values of the 3D seismic data approximated by PHLCT and DCT, where the size of each block is also $8 \times 8 \times 8$. The PSNR values of PHLCT are 0.2 dB higher than those of DCT. 3D PHLCT also performs better than 3D DCT.

Table 2: PSNR values (dB) of 3D real seismic data approximated by 3D PHLCT and 3D DCT

Ratio of retained coefficients	1.0%	1.5%	2.0%	2.5%
PHLCT	15.4084	16.5051	17.2654	17.8589
DCT	15.1489	16.2813	17.0713	17.6881



(a) Slice A (b) Slice B (c) Slice C

Figure 1. 2D slices from 3D seismic data.

Finally, we will compare 2D PHLCT with 3D PHLCT. From the different directions, we take three slices from 3D seismic data and approximate them by 2D PHLCT. PSNR values are shown in Table 3. From this, we find that the PSNR value of Slice A is higher than that of Slice B or Slice C. This is because Slice A is much smoother than Slice B and Slice C. Comparing Table 3 with Table 2, we find that for the same rate of the retained coefficients, the PSNR values of 3D PHLCT are higher than that of 2D PHLCT, so 3D PHLCT performs better than 2D PHLCT when we approximate 3D seismic data. This is also true in approximation theory.[9]

Table 3: PSNR values (dB) of slices of 3D real seismic data approximated by 2D PHLCT

Ratio of retained coefficients	1.0%	1.5%	2.0%	2.5%
Slice A	13.4705	14.0368	15.5340	16.2911
Slice B	7.9232	7.9559	9.7533	11.0347
Slice C	8.2771	8.3082	10.1260	11.4538

From these tables, we see that high-dimensional PHLCT algorithm can approximate high-dimensional data more efficiently than DCT algorithm and 2D PHLCT algorithm.

ACKNOWLEDGMENTS

This research was partially supported by NSF grant DMS-0410406 and ONR grant N00014-07-1-0166.

REFERENCES

1. K. Yamatani and N. Saito, “Improvement of DCT-based compression algorithms using Poisson’s equation,” *IEEE Trans. Image Proc.*, **15**, pp. 3672–3689, 2006.
2. N. Saito and J. F. Remy, “The polyharmonic local sine transform: a new tool for local image analysis and syntheses without edge effect,” *Appl. Comput. Harmonic Anal.*, **20**, pp. 41–73, 2006.
3. A. Averbuch, M. Israeli, and L. Vozovoi, “A fast Poisson solver of arbitrary order accuracy in rectangular regions,” *SIAM J. Sci. Comput.*, **19**, pp. 933–952, 1998.
4. E. Braverman, M. Israeli, A. Averbuch, and L. Vozovoi, “A fast 3D Poisson solver of arbitrary order accuracy,” *J. Comp. Physics*, **144**, pp. 109–136, 1998.
5. G. Strang, “The discrete cosine transform,” *SIAM Rev.*, **41**, pp. 135–147, 1999.
6. W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*. New York: Van Nostrand Reinhold, 1993.
7. R. J. Clarke, “Relation between the Karhunen-Loève and cosine transforms,” *IEE Proc., Part F*, **128**, pp. 359–360, 1981.

8. K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, and Applications*. San Diego, CA: Academic Press, 1990.
9. R. A. DeVore, “Non-linear approximation” in *Acta Numerica*, pp. 51–150, Cambridge Univ. Press, 1998.