Orthogonal Directional Transforms using Discrete Directional Laplacian Eigen Solutions for Beyond HEVC Intra Coding

Itsik Dvir, *Senior Member, IEEE,* Dror Irony, David Drezner, Ady Ecker, Amiram Allouche, and Natan Peterfreund

***Abstract*—We introduce new transforms for efficient compression of image blocks with directional preferences. Each transform, which is an orthogonal basis for a specific direction, is constructed from an eigen-decomposition of a discrete directional Laplacian system matrix. The method is a natural extension of the DCT, expressing the Laplacian in Cartesian coordinates rotated to some predetermined angles. Symmetry properties of the transforms over square domains lead to efficient computation and compact storage of the directional transforms. A version of the directional transforms was implemented within the beyond HEVC software and demonstrated significant improvement for intra block coding.**

***Index Terms*—video coding, directional bases, Laplace equations, discrete cosine transforms.**

I. Introduction

THE two-dimensional discrete cosine transform (2D-DCT) is the most widely used unitary transform in image and video coding standards, such as JPEG, MPEG-2, MPEG-4, AVC/H.264, and HEVC/H.265. The DCT is an orthogonal and separable basis set of cosine functions, used to represent image information of rectangular blocks by spatial frequency coefficients. Generally, the energy of smooth signals, as in typical images, is concentrated in the low frequency coefficients. Efficient compression is obtained when most of the energy is concentrated in a small number of quantized coefficients.

The 1D-DCT, discovered in 1974 [1] [2], was derived as an approximation to the eigenvectors of the covariance matrix of a class of stationary Markovian signals. These eigenvectors are an approximation to an optimal Karhunen-Loeve basis for compressing this class of signals. Thus, the analytical 1D-DCT vectors are close to optimal. It is well known that each one of the eight types of 1D-DCT is an orthogonal basis for a discrete interval. Strang [3] derived the 1D-DCT basis in a different way, by showing that each DCT type contains the eigenvectors of a second difference symmetric matrix. The eigenvectors are determined by the Boundary Conditions (BC) at each endpoint of the interval (Dirichlet or Neumann), whether the BC is applied at a grid point or at the middle between two adjacent grid points, and whether the eigenfunctions have an even or odd extensions around that point. Continuity at the interval boundaries made DCT-II more attractive and the transform of choice in image and video coding standards. In practice, the 2D-DCT coefficients are computed by fast and efficient schemes applied separably along the vertical and horizontal directions.

Instead of applying the DCT directly to image blocks, modern encoding frameworks, such as HEVC, apply the DCT to residual blocks. The residual blocks are obtained by subtracting prediction blocks from the original image blocks. The purpose of prediction blocks is to exploit information that was already transmitted. Prediction blocks can be generated either from other frames (inter-prediction) or other blocks in the same frame (intra-prediction).

The angular intra-prediction mechanism of HEVC [4] replicates boundary pixels along straight lines. Directional patterns may still remain in the residual blocks, as shown in Fig. 1, for several reasons. First, angular prediction might not remove all the directional structures in the original image blocks, even after increasing the number of intra-prediction angular directions to 33 in the HEVC standard [5] and 65 directions in proposals for beyond HEVC [6]. Second, whenever there are discontinuities along the edges of a predicted block, angular prediction might actually insert directional stripes that did not exist in the original block.

The 2D-DCT is a separable basis, and deals relatively well with horizontal or vertical patterns. On the other hand, oriented patterns are typically characterized by high frequencies in one direction and low frequencies in the orthogonal direction. Discontinuities along the horizontal and vertical axes may increase the number of non-zero coefficients at high frequencies due to Gibbs phenomenon. As a result, the representation of oriented residual blocks by DCT might be inefficient.

To address this shortcoming, several attempts were made in the last decade [7]–[23] to develop directional transforms, i.e. transforms that are better tuned to particular orientations. Three categories of directional transforms for image coding have been overviewed in [7]: (i) reorganization of pixels [8], [9]; (ii) lifting [10], [11]; and (iii) data-dependent directional transforms [12].

In reorganization based approaches [8], block's pixels are reorganized according to a selected direction. Conventional 1D transforms of variable lengths are applied. The coefficients produced by the directional transforms are rearranged and a second pass of 1D transforms is applied to the coefficients. These methods are suboptimal since the transformation is not orthogonal in 2D, and the 2D relationships between the

This research was conducted at Huawei Technologies Co., Ltd. Tel Aviv Research Centre (TRC), Toga Networks, 4 Ha'Harash St., Hod Hasharon, Israel, 457706. e-mail: itsik.dvir@gmail.com.

Manuscript received July 2, 2019; revised January 20, 2020.

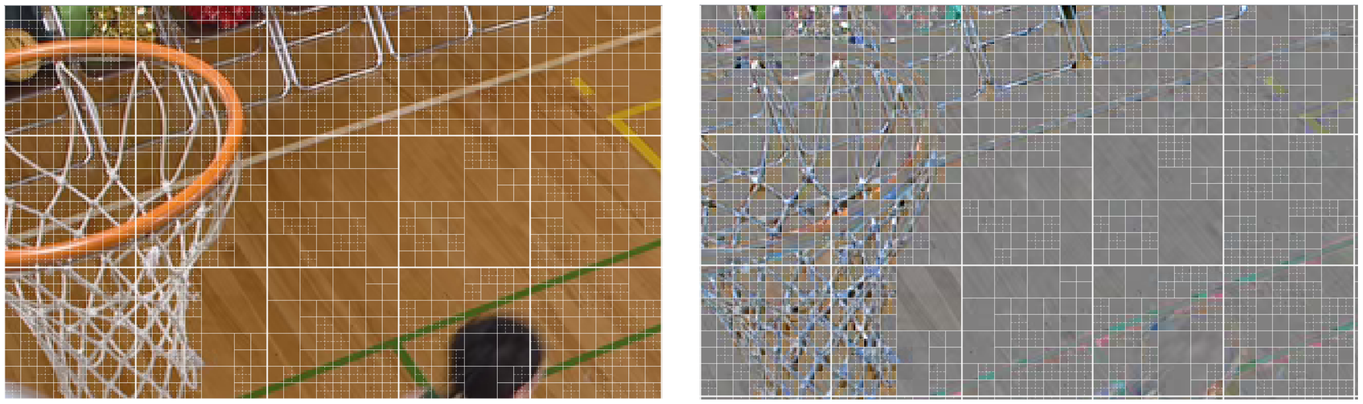


Fig. 1: Left: Reconstructed part from the BasketballDrill sequence by standard HEVC decoder. The intra-prediction quadtree is overlaid on the image. Right: The residual image, after subtraction of the intra-prediction. Note that strong directional structures remain in the residual even after angular prediction. The standard DCT is efficient at representing smooth blocks, or blocks with horizontal or vertical structures, but is less efficient at representing directional structures.

original pixels are distorted.

Lifting-based methods [10], [11] also apply 1D transforms, but they use 2D image interpolations. Similarly to pixel reorganization methods, they have inefficiencies associated with the fact that oriented lines at the center and the corners of a square block are of different lengths.

Data-dependent transforms are based on training data in the spirit of the Karhunen-Loeve transform (KLT). In [12], mode-dependent directional transforms (MDDT) are derived from KLT using prediction residuals from training video data. Signal Dependent Transform (SDT) is used for various block sizes [13].

The directional transforms we will develop in this paper fit in a recent trend where the codec can use additional transforms that complement the DCT. From a sparse coding perspective, the multiple bases are essentially an over-complete dictionary [24]. Obviously, larger dictionaries can lead to sparser representations. On the other hand, over-complete representations require signaling which basis to use. Both the signaling overhead and the search time can be saved if the transform can be linked to the intra-prediction mode. The DCT/DST mode-dependent combination [14] and the Explicit Multiple Transform (EMT) methods [15] use this approach. EMT uses selected transforms from the DCT/DST families (DST-VII, DCT-VIII, DST-I, and DCT-V) in addition to the current DCT-II transform in HEVC. Similarly, Dvir et al. [23] used the standard DCT in a framework where quadtree blocks can be split diagonally. This can deal better with directional cuts between regions, but not with general directional patterns.

Another recent approach uses a secondary transform applied to the coefficients of the conventional 2D-DCT/DST. Among the secondary transforms are the Rotational Transform (ROT) [16], which uses Givens rotation matrices, and the mode dependent non-separable secondary transform (MDNSST) [17], which is applied to the lower frequencies $4{\times}4$ and $8{\times}8$ groups of transform coefficients. A similar idea that rotates pairs of 2D-DCT vectors with the same eigenvalues in their 2D subspace was suggested in [21]. While these transforms are orthogonal, the rotations occur in a high-dimensional space. The resulting basis vectors do not necessarily suit to represent rotations in the image plane.

Selesnick and Guleryuz [20] introduced two methods for generating directional orthogonal bases. Their first method is based on enforcing diagonal structure on a lower-dimentionality system matrix, and a completion to a basis. Their second method takes the eigen-basis of a system matrix that is constructed from oriented first-derivative terms. This differs from the DCT which is based on second-derivative terms.

In this paper we introduce a new scheme for the generation of directional transforms, which are orthogonal in 2D and form a complete basis set for each direction. These transforms may be used for efficient representation of image blocks with directional content. The scheme is demonstrated and evaluated using the beyond HEVC codec.

The paper is organized as follows. In Section II, we introduce the continuous and discrete directional Laplacian operator. In Section III, we present the discrete directional Laplacian eigen-basis for the $N{\times}N$ square. The computation of transform coefficients (forward transform) for a given image block and the reconstruction by the inverse transform are described in Section IV. Implementation details are discussed in Section V. The efficiency of the proposed representation is demonstrated in Section VI through evaluation on the Common Test Conditions (CTC) sequences [25]. Finally, conclusions are given in Section VII.

II. The directional Laplacian operator

A. The continuous directional Laplacian

The two-dimensional continuous *Laplacian operator* is defined by

$$\Delta f = \nabla^2 f = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) f \ . \qquad (1)$$

The Laplacian in (1) can be written as a decomposition of parallel and perpendicular parts in relation to any angle θ:

$$\Delta f = \nabla^2 f = \nabla^2_{\parallel} f + \nabla^2_{\perp} f \ . \qquad (2)$$

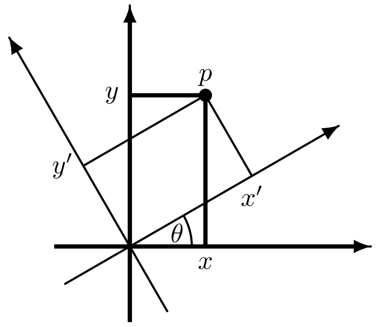


Fig. 2: Rotated coordinates system. The point $p=(x,y)$ has coordinates (x',y') in the rotated system.

In the rest of the paper, we will refer to $\nabla^2_{\parallel} f$ as the *directional Laplacian*.

For completeness of the presentation we provide below the derivation from Broadhead [26], who used it for anisotropic filtering. Let (x,y) be the Cartesian coordinates of a point with respect to a first coordinate system and (x',y') the Cartesian coordinates of that point with respect to a second coordinates system that is rotated by an angle θ, as shown in Fig. 2. Accordingly, (x,y) and (x',y') are related to each other as:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} \ . \tag{3}$$

To express the Laplacian in the rotated coordinates (x',y'), we write the first derivative for x'

$$\frac{\partial f}{\partial x'} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial x'} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial x'} = \frac{\partial f}{\partial x}\cos(\theta) + \frac{\partial f}{\partial y}\sin(\theta) \ . \tag{4}$$

Then, the second derivative for x' is calculated by applying (4) a second time:

$$\begin{aligned} \frac{\partial^2 f}{\partial x'^2} &= \left(\cos(\theta)\frac{\partial}{\partial x} + \sin(\theta)\frac{\partial}{\partial y}\right)\left(\cos(\theta)\frac{\partial f}{\partial x} + \sin(\theta)\frac{\partial f}{\partial y}\right) \\ &= \cos^2(\theta)\frac{\partial^2 f}{\partial x^2} + 2\cos(\theta)\sin(\theta)\frac{\partial^2 f}{\partial x \partial y} + \sin^2(\theta)\frac{\partial^2 f}{\partial y^2} \ . \end{aligned} \tag{5}$$

Similarly, the first and second derivatives for y' are given by:

$$\frac{\partial f}{\partial y'} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial y'} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial y'} = \frac{-\partial f}{\partial x}\sin(\theta) + \frac{\partial f}{\partial y}\cos(\theta) \ . \tag{6}$$

$$\begin{aligned} \frac{\partial^2 f}{\partial y'^2} &= \\ &= \left(-\sin(\theta)\frac{\partial}{\partial x} + \cos(\theta)\frac{\partial}{\partial y}\right)\left(-\sin(\theta)\frac{\partial f}{\partial x} + \cos(\theta)\frac{\partial f}{\partial y}\right) \\ &= \sin^2(\theta)\frac{\partial^2 f}{\partial x^2} - 2\cos(\theta)\sin(\theta)\frac{\partial^2 f}{\partial x \partial y} + \cos^2(\theta)\frac{\partial^2 f}{\partial y^2} \ . \end{aligned} \tag{7}$$

Referring to x' and y' as the parallel and perpendicular directions, the parallel and the perpendicular parts of the Laplacian are given in (5) and (7), respectively:

$$\nabla^2_{\parallel} f = \frac{\partial^2 f}{\partial x'^2} \ , \tag{8}$$

$$\nabla^2_{\perp} f = \frac{\partial^2 f}{\partial y'^2} \ . \tag{9}$$

Note that the identity in (2) is obtained by summing (5) and (7) for any angle θ:

$$\nabla^2_{\parallel} f + \nabla^2_{\perp} f = \frac{\partial^2 f}{\partial x'^2} + \frac{\partial^2 f}{\partial y'^2} = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = \nabla^2 f = \Delta f \ . \tag{10}$$

Although we focus mainly on the two-dimensional case, it is instructional to derive the directional Laplacian more generally in any number of dimensions. Let $\mathbf{R}$ be an n-dimensional orthogonal matrix that aligns the first coordinate with some desired direction in the coordinates transformation

$$\mathbf{x}' = \mathbf{R}\mathbf{x} \ . \tag{11}$$

For example, in two dimensions $\mathbf{R} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$. The Hessian matrix operator $\mathbf{H} = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right]_{i,j=1,\dots,n}$ in the rotated coordinates system is

$$\mathbf{H}' = \mathbf{R}\mathbf{H}\mathbf{R}^{\mathbf{T}} \ . \tag{12}$$

Formula (12) is just a change of basis formula for the linear operator $\mathbf{H}$ [27]. The directional Laplacian in the direction of the first rotated coordinate is

$$\frac{\partial^2 f}{\partial {x'_1}^2} = \mathbf{R_{row1}}\mathbf{H}\mathbf{R^T_{row1}} \ , \tag{13}$$

where $\mathbf{R_{row1}}$ is the first row of $\mathbf{R}$. For example, in two dimensions $\mathbf{R_{row1}} = [\cos(\theta) \ \ \sin(\theta)]$ and (13) agrees with (5).

B. The discrete directional Laplacian

In this section we present our discrete approximation to the continuous directional Laplacian operators (5) and (7). We apply the finite difference method [28] for a regularly sampled image block.

Denote $x_i = ih$, $y_j = jh$, $f(i,j) = f(x_i,y_j)$, where $i,j = 0,1,2,\dots,N-1$, $h = 1/N$, and N is the block size. Using the central difference and the Taylor series expansion about the point (x_i,y_j) for its neighboring points, we get:

$$\begin{aligned} \left.\frac{\partial^2 f}{\partial x \partial y}\right|_{i,j} &= \frac{1}{4h^2}(f(i+1,j+1) - f(i+1,j-1) \\ &\qquad - f(i-1,j+1) + f(i-1,j-1)) + O(h^2) \ , \\ \left.\frac{\partial^2 f}{\partial x^2}\right|_{i,j} &= \frac{1}{h^2}(f(i+1,j) - 2f(i,j) + f(i-1,j)) + O(h^2) \ , \\ \left.\frac{\partial^2 f}{\partial y^2}\right|_{i,j} &= \frac{1}{h^2}(f(i,j+1) - 2f(i,j) + f(i,j-1)) + O(h^2) \ . \end{aligned} \tag{14}$$

In the formula above, $O(h^2)$ stands for error terms of order h^2 or higher which are neglected.

The expression for the parallel Laplacian is obtained by substituting (14) in (5):

$$\begin{aligned} \nabla^2_{\parallel} f = \frac{1}{4h^2}\big[&-8f(i,j) \\ &+ 4\cos^2(\theta)(f(i-1,j) + f(i+1,j)) \\ &+ 4\sin^2(\theta)(f(i,j-1) + f(i,j+1)) \\ &+ \sin(2\theta)(f(i+1,j+1) - f(i+1,j-1) \\ &\qquad - f(i-1,j+1) + f(i-1,j-1))\big] \ . \end{aligned} \tag{15}$$

The stencil for the parallel component of the Laplacian can be written in matrix form:

$$\nabla^2_{\parallel} = \frac{1}{4h^2}\begin{bmatrix} -\sin(2\theta) & 4\sin^2(\theta) & \sin(2\theta) \\ 4\cos^2(\theta) & -8 & 4\cos^2(\theta) \\ \sin(2\theta) & 4\sin^2(\theta) & -\sin(2\theta) \end{bmatrix} . \qquad (16)$$

The stencil for the perpendicular component can be derived similarly:

$$\nabla^2_{\perp} = \frac{1}{4h^2}\begin{bmatrix} \sin(2\theta) & 4\cos^2(\theta) & -\sin(2\theta) \\ 4\sin^2(\theta) & -8 & 4\sin^2(\theta) \\ -\sin(2\theta) & 4\cos^2(\theta) & \sin(2\theta) \end{bmatrix} . \qquad (17)$$

The standard five-point stencil for the Laplacian [28] is obtained by summing (16) and (17), in accordance with (2):

$$\nabla^2_{\parallel} + \nabla^2_{\perp} = \nabla^2_5 = \frac{1}{h^2}\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} . \qquad (18)$$

Some examples are given below for special rotation angles: (i) for $\theta = 0$ we get the discrete 1D Laplacian (second derivative) operator:

$$\nabla^2_{\parallel} = \frac{1}{h^2}\begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} \;,\; \nabla^2_{\perp} = \frac{1}{h^2}\begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad (19)$$

(ii) for $\theta = \pi/4$:

$$\nabla^2_{\parallel} = \frac{1}{4h^2}\begin{bmatrix} -1 & 2 & 1 \\ 2 & -8 & 2 \\ 1 & 2 & -1 \end{bmatrix} , \nabla^2_{\perp} = \frac{1}{4h^2}\begin{bmatrix} 1 & 2 & -1 \\ 2 & -8 & 2 \\ -1 & 2 & 1 \end{bmatrix} . \qquad (20)$$

III. The directional Laplacian eigenfunctions

After defining the directional Laplacian operator, in this section we develop directional orthogonal bases for square blocks. Each basis is made of the eigenfunctions of an eigen-problem generated by applying the directional Laplacian operator for some angle θ at each point in the square.

System matrix specification. The eigen-problem we construct is of the form $\mathbf{A}\mathbf{v}{=}\lambda\mathbf{v}$. $\mathbf{v}$ is an $N^2{\times}1$ vector representing an $N{\times}N$ block of pixels in a column-major order. The $N^2{\times}N^2$ system matrix $\mathbf{A}$ is constructed such that its k-th row applies the discrete directional Laplacian operator (16) at the k-th pixel in our column-major order.

Boundary specification. The Laplacian stencil (16) cannot be applied directly at boundary pixels of the square domain. There are many possible ways to define boundary conditions for the directional Laplacian operator, and each will result in a different basis of eigenfunctions. It is beyond the scope of this work to examine the effect of different boundary conditions. We implemented one of the simplest rules possible: we keep all the coefficients inside the support of the image, except the central one, and adjust the central coefficient so that the sum of the stencil is 0. For example, the stencils for the top row and top-left corner of the block are:

$$\nabla^2_{\parallel_{\text{top}}} = \frac{1}{4h^2}\begin{bmatrix} 4\cos^2(\theta) & -4-4\cos^2(\theta) & 4\cos^2(\theta) \\ \sin(2\theta) & 4\sin^2(\theta) & -\sin(2\theta) \end{bmatrix}$$

$$\nabla^2_{\parallel_{\text{top-left}}} = \frac{1}{4h^2}\begin{bmatrix} -4+\sin(2\theta) & 4\cos^2(\theta) \\ 4\sin^2(\theta) & -\sin(2\theta) \end{bmatrix} . \qquad (21)$$

The stencils for the other boundary lines and corners are derived similarly.

As an example, the system matrix for a $4{\times}4$ block with $\theta = \pi/4$ is

$$\begin{bmatrix}
-3 & 2 & 0 & 0 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & -6 & 2 & 0 & 1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2 & -6 & 2 & 0 & 1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 2 & -5 & 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 1 & 0 & 0 & -6 & 2 & 0 & 0 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 2 & 1 & 0 & 2 & -8 & 2 & 0 & 1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 2 & 1 & 0 & 2 & -8 & 2 & 0 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 2 & 0 & 0 & 2 & -6 & 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 1 & 0 & 0 & -6 & 2 & 0 & 0 & 2 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 2 & 1 & 0 & 2 & -8 & 2 & 0 & 1 & 2 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 2 & 1 & 0 & 2 & -8 & 2 & 0 & 1 & 2 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & 0 & 0 & 2 & -6 & 0 & 0 & 1 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 0 & 0 & -5 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & 1 & 0 & 2 & -6 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & 1 & 0 & 2 & -6 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & 0 & 0 & 2 & -3
\end{bmatrix} . \qquad (22)$$

Transform specification. The 2D orthogonal directional basis for the domain is generated by computing the eigenvalues λ_k and eigenvectors $\mathbf{v}_k$ of the system matrix $\mathbf{A}$, which is symmetric and real

$$\mathbf{A}_k = \lambda_k \mathbf{v}_k \ , \ k = 1, 2, \ldots, N^2 \ . \qquad (23)$$

The eigenvectors of $\mathbf{A}$ form an orthogonal basis of the image over the square block:

$$\mathbf{V}\mathbf{V}^T = \mathbf{V}^T\mathbf{V} = \mathbf{I}_{N^2\times N^2} \qquad (24)$$

Here, $\mathbf{v}_k$ is the k-th column of the matrix $\mathbf{V}$, and $\mathbf{I}$ is the $N^2{\times}N^2$ identity matrix. The eigen-decomposition of $\mathbf{A}$ is done offline by numerical software and stored for use in the transform phase.

The eigenvectors are sorted according to the order of the absolute values of the eigenvalues, with the first (smallest) eigenvalue corresponding to the lowest spatial "frequency"

$$0 = |\lambda_1| < |\lambda_2| \le |\lambda_3| \le \cdots \le |\lambda_{N^2}| \ . \qquad (25)$$

The resulting eigenfunctions are illustrated in Fig. 3 and Fig. 4. Note how the directional Laplacian operator induces directional 2D bases with directional preferences.

IV. Forward and inverse transforms

In this section the forward transformation and the inverse transformation (reconstruction) are described.

A. Forward directional transform

Given the samples of the image block, an $N^2{\times}1$ vector $\mathbf{p}$ is constructed from the image samples using the same column-major scanning order used to generate the system matrix $\mathbf{A}$.

The k-th transform coefficient, $\mathbf{c}_k$, is obtained as an inner product of the corresponding eigenvector $\mathbf{v}_k$ and the vector of the image samples $\mathbf{p}$:

$$\mathbf{c}_k = \mathbf{v}^T\mathbf{p} \ . \qquad (26)$$

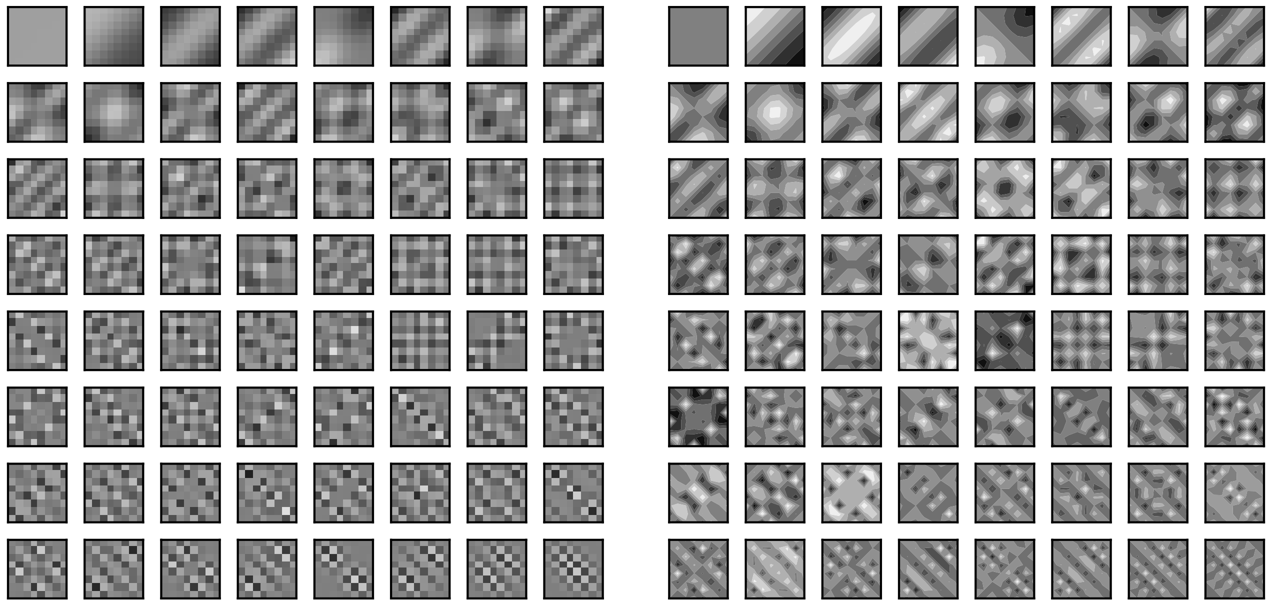


Fig. 3: Left: The 64 eigen-basis for $8{\times}8$ blocks and $\theta{=}\pi/4$. The basis functions are ordered according to the magnitude of the eigenvalues, from the top-left corner by rows. The first vector is a constant (DC). 0.5 was added to the functions for visualization. Note that each eigenfunction $\mathbf{v}$ is either symmetric, $\mathbf{v}_k(i,j) = \mathbf{v}_k(N{-}j, N{-}i)$, or anti-symmetric, $\mathbf{v}_k(i,j) = -\mathbf{v}_k(N{-}j, N{-}i)$. Right: Contour plots of the eigenfunctions. This plot highlights the orientations of the basis functions and their symmetric or anti-symmetric hills and valleys.

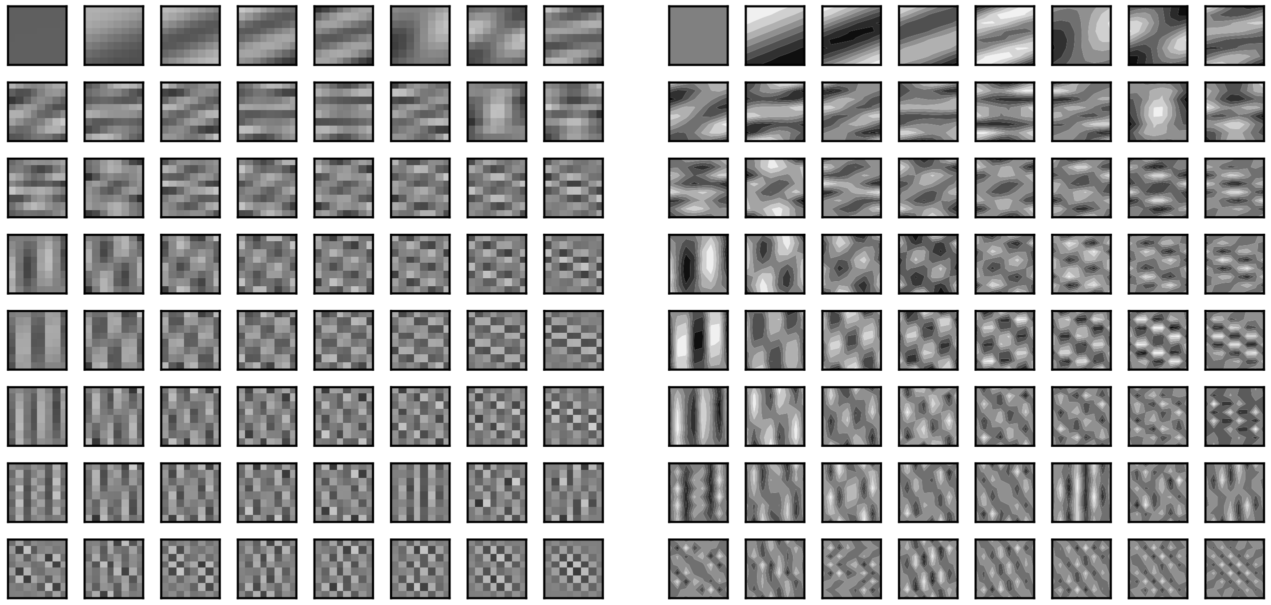


Fig. 4: Left: The 64 eigen-basis for $8{\times}8$ blocks and $\theta{=}\pi/8$. Right: Contour plots of the eigenfunctions.

In matrix notation the vector of coefficients $\mathbf{c}$ is:

$$\mathbf{c} = \mathbf{V}^T \mathbf{p} \ , \tag{27}$$

where the matrix $\mathbf{V}$ is an orthonormal directional transformation from the image domain to the spectral domain.

Since the eigenvectors are sorted, for a directional image block $\mathbf{p}$ we expect $\|\mathbf{Ap}\|/\|\mathbf{p}\|$ to be relatively small, because it is the application of the directional Laplacian operator to a directional pattern. Since the first eigenvectors of $\mathbf{A}$ span the subspace of small $\|\mathbf{Au}\|/\|\mathbf{u}\|$, it is expected that the first coefficients in (27) would be relatively large and the last ones relatively small. After coefficients quantization, many of the last coefficients are expected to become zero. Codecs like HEVC use this property for efficient encoding of the transform coefficients, e.g. by signaling the last significant coefficient and signaling coefficient groups of all zero coefficients [29].

B. Inverse directional transform

The inverse transform produces the reconstructed vector $\mathbf{p}_{\text{rec}}$, defined as a linear combination of the eigenvectors $\mathbf{v}_k$ weighted by the coefficients:

$$\mathbf{p}_{\text{rec}} = \sum_{k=1}^{N^2} c_k \mathbf{v}_k = \mathbf{Vc} \tag{28}$$

In matrix notation the reconstructed vector is obtained by the inverse transformation matrix $\mathbf{V}$ applied to the coefficients vector $\mathbf{c}$. Note that by using (27) and (24), the reconstruction is exact, since $\mathbf{VV}^T = \mathbf{I}$:

$$\mathbf{p}_{\text{rec}} = \mathbf{Vc} = \mathbf{VV}^T\mathbf{p} = \mathbf{p} \ . \tag{29}$$

V. Implementation

A. Implementation in the beyond HEVC codec

After describing the directional Laplacian orthogonal transforms, in this section we describe our implementation. We evaluated the effectiveness of the transform using the HM-16.6-KTA-2.0 (referred here as KTA-2.0) [30], with its associated Common Test Conditions (CTC) video sequences. KTA-2.0 is an intermediate research version set by ITU-T VCEG towards a new standard beyond HEVC [6].

We implemented the directional Laplacian transform (abbreviated "DirLap") as an additional transform candidate for KTA-2.0's EMT method [15] in the Transform Unit (TU) of intra blocks. DirLap was evaluated for all blocks up to 32×32 pixels, except blocks with DC or planar intra-prediction, because these blocks have smaller chances to contain directional structures. Similarly to the three EMT transform candidates, DirLap is tested in a rate-distortion sense.

In principle, the encoder can test the encoding of each block with multiple directional transforms, for a range of angles θ, and select the best transform. This approach is computationally very expensive, and adds the overhead of signaling which transform was selected. Our implementation avoids this by relying on HEVC's angular intra-prediction [4]. For each block, we test at most one directional transform, according to the selected intra-prediction angle. Hence, there is no need to send a special signal for the direction of the transform.

The set of transform matrices (for each intra-prediction direction) are stored both at the encoder and decoder sides, so there is no need to send the transform matrices with the encoded bitsteam. The transform matrices are integer matrices, obtained by rounding each matrix component to 14 bits (including a sign bit). This representation was chosen to achieve small rounding errors in transforms of size 32×32 for 8-bits videos. For hardware designs, the number of representation bits could be further reduced for smaller block sizes. For our software implementation, it was more convenient to use these 14 significant bits numbers inside 16 bits numbers.

The encoder computes the transform coefficients for the forward DirLap from the residual samples using (27). Both the encoder and the decoder apply the inverse transform (28) to compute the reconstructed samples from the quantized coefficients. On both sides, the same column-major scanning order is used to convert between $N^2{\times}1$ vectors and $N{\times}N$ blocks of pixels.

Coefficient scaling, quantization and scanning: The standard coefficient scaling and quantization scheme used by the HEVC [31] is also applied in the DirLap case. The same scaling factors are used for the quantized coefficients to fit within 16 bits for 8-bits videos. However, the scanning order of the DirLap coefficients is according to increasing magnitude of the eigenvalues.

Signaling contexts: HEVC uses Context-Adaptive Binary Arithmetic Coding (CABAC) to adapt the encoding of certain bits to their varying probabilities in the video stream. We added several such contexts to adapt to the occurrence probabilities of the directional transforms. A new context flag (EmtDirFlag) is added to the bit-stream for signaling whether DirLap or another one of KTA-2.0's three EMT transforms is used. The flag is included in the bit-stream only when DirLap is a possible transform candidate, i.e. neither planar nor DC intra-prediction modes were chosen. As the occurrence probabilities of DirLap and EMT may be different, we added a separate context (EmtCuFlagSCModel) to EmtCuFlag, which signals in KTA-2.0 at the Coding Unit (CU) level whether EMT is used for Transform Units (TU) of the current CU. Similarly, we added a separate context (EmtTuIdxSCMode) to EmtTuIdx, which signals at the TU level which transform index of the three EMT transform candidates is used. In addition, separate contexts are added for coefficients coding (SigCoeffGroupSCModel, SigSCModel, LastX, LastY, and OneSCModel).

B. Memory requirements reduction

Unlike the DCT, the directional Laplacian is non-separable, because the directional Laplacian operator contains cross derivatives for angles other than 0° or 90°. Therefore, a naive implementation needs to store full transform matrices of size $N^2{\times}N^2$ for $N = 4, 8, 16, 32$. In KTA-2.0 [6], there are 65 angular intra-prediction directions (numbered $d{=}2, \ldots, 66$). This implies that the memory requirement of a naive implementation is significant. However, the memory requirement can be significantly reduced to a manageable size by exploiting the following properties:

TABLE I: Each intra-prediction mode has a symmetric mode in the range $2, \dots, 18$. We re-use each transform matrix for this range four times. Mode 18 is horizontal and requires 1D transform, hence only 16 full transforms need to be stored.

Prediction mode	Symmetry	Symmetric mode
$d = 2, \dots, 18$	Original	d
$d = 19, \dots, 34$	Reflection along the horizontal axis	$36{-}d$
$d = 35, \dots, 50$	90° rotation	$d{-}32$
$d = 51, \dots, 66$	Reflection along the diagonal axis	$68{-}d$

1) *1D transform:* For the horizontal (d=18) and vertical (d=50) directions, the transform matrix is built from 1D transforms, applied to each one of the N columns or N rows. The matrix for d=50 is a rotated version of the matrix for d=18. Therefore, only N eigenvectors of size N are stored.
2) *Angular symmetries:* Each directional transform matrix can be used to generate the transforms for 3 additional directions. We can apply a basic symmetry of the square to the transform matrix, e.g. a reflection along the horizontal axis, a 90° rotation, or a reflection along the diagonal axis (transpose). The transformed matrix can then multiply a vectorized image block. Essentially, this is equivalent to a change in the scanning order of the pixels in the block being vectorized before multiplying by the original transform matrix. Hence, only a subset of 16 out of the 65 matrices needs to be stored for each block size, as shown in Table I and Fig. 5.
3) *Symmetric and anti-symmetric eigenvectors:* An eigenvector of a directional transform matrix, seen as a square, is either symmetric, $\mathbf{v}_k(i,j) = \mathbf{v}_k(N{-}j, N{-}i)$, or anti-symmetric, $\mathbf{v}_k(i,j) = -\mathbf{v}_k(N{-}j, N{-}i)$, as shown in Fig. 3. The symmetry here is with respect to 180° rotation around the center of the $N{\times}N$ block, $i, j = 0, 1, \dots, N-1$. Since one-half $(N{\times}N/2)$ of the eigenvectors are symmetric, and the other half is anti-symmetric, only the first one-half entries in the scanning order of each eigenvector needs to be stored. These properties were verified for all directions and all sizes of the integer transform matrices. In addition, 1 bit per eigenvector is stored to indicate whether it is symmetric or anti-symmetric.
4) *The DC eigenvector:* The first eigenvector is a constant function. Only one value needs to be stored for this eigenvector.
5) *Large blocks:* Transforms of large blocks are selected mostly for smooth, low-frequency, blocks. high-frequency blocks are often split by the quad-tree search mechanism. To save memory and computation time, we use only the first 256 vectors for 32×32 blocks.

By employing the properties above, less than $1/8$ of the original memory requirement is needed. Table II summarizes the total memory requirement for the 16 transform matrices, from which the transforms of all 65 directions can be generated.

$$\boxed{\begin{array}{cccc} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{array}} \qquad \boxed{\begin{array}{cccc} 4 & 8 & 12 & 16 \\ 3 & 7 & 11 & 15 \\ 2 & 6 & 10 & 14 \\ 1 & 5 & 9 & 13 \end{array}}$$

(a) (b)

$$\boxed{\begin{array}{cccc} 4 & 3 & 2 & 1 \\ 8 & 7 & 6 & 5 \\ 12 & 11 & 10 & 9 \\ 16 & 15 & 14 & 13 \end{array}} \qquad \boxed{\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{array}}$$

(c) (d)

Fig. 5: To save memory, transformation matrices are re-used by four symmetric prediction directions. We illustrate above the four matrices that can be generated from a single matrix for a 4×4 block. (a) matrix elements in column-major order. (b) position of the elements after horizontal reflection. (c) position after a 90° rotation. (d) position after reflection along the diagonal. In the implementation, these four matrices are never generated. The block that multiplies a matrix is vectorized according to the desired scanning order. Similar operations take place for the inverse transform.

TABLE II: Memory requirements for the 16 transform matrices stored by DirLap. All 65 directional transforms can be generated on-the-fly from 16 transform matrices. If only 256 eigenvectors are stored for 32×32 blocks, the total memory is 5,172 KB.

Block size	256 eigenvectors	All eigenvectors
4	4 KB	4 KB
8	64 KB	64 KB
16	1,022 KB	1,022 KB
32	4,082 KB	16,370 KB
Total	5,172 KB	17,460 KB

C. Running time reduction

The computation of the directional transforms is more expensive than the DCT. Unlike the DCT, our directional transform is not separable and we don't have an FFT-type implementation for it. Still, the running time can be significantly reduced.

For a symmetric eigenvector, the elements of the dot product with the block are of the form $\mathbf{v}_k(i,j) \cdot \mathbf{B}(i,j) + \mathbf{v}_k(N{-}j, N{-}i) \cdot \mathbf{B}(N{-}j, N{-}i) = \mathbf{v}_k(i,j) \cdot [\mathbf{B}(i,j) + \mathbf{B}(N{-}j, N{-}i)]$. For an anti-symmetric eigenvector, the elements are of the form $\mathbf{v}_k(i,j) \cdot \mathbf{B}(i,j) + \mathbf{v}_k(N{-}j, N{-}i) \cdot \mathbf{B}(N{-}j, N{-}i) = \mathbf{v}_k(i,j) \cdot [\mathbf{B}(i,j) - \mathbf{B}(N{-}j, N{-}i)]$. Using the symmetry or anti-symmetry property of each eigenvector in the transform matrix, a 50% reduction in the number of multiplications is obtained.

In addition, the KTA-2.0 implementation contains Single Instruction Multiple Data (SIMD) instructions. In our implementation of the directional Laplacian transform, we used SIMD instructions to execute multiplications in parallel. Two 16 bits numbers are multiplied simultaneously utilizing 32 bits out of the 128 bits used in SIMD. This gives a further reduction

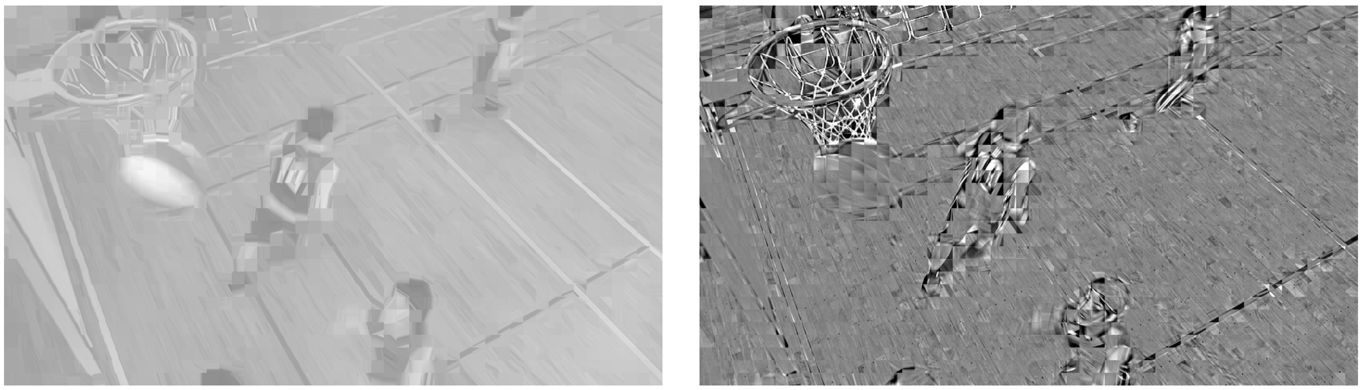


Fig. 6: Left: Prediction of 16×16 blocks form the BasketballDrill sequence. Right: Residual 16×16 blocks, contrast-enhanced for visualization. Note again the linear structures remaining after angular intra-prediction.

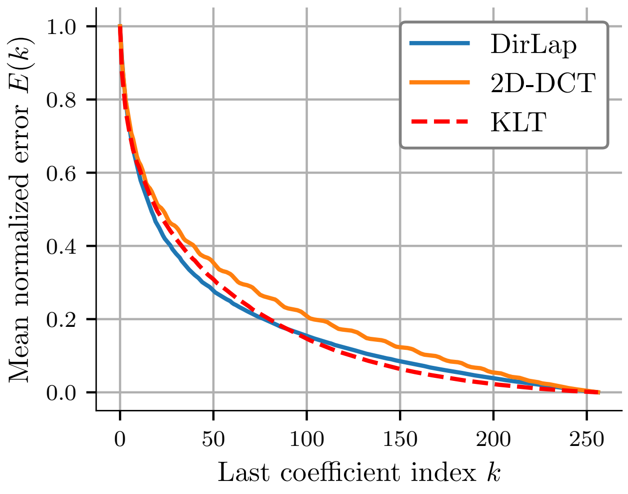


Fig. 7: Mean normalized energy error $E(k)$ of DirLap, 2D-DCT, KLT, vs. the number of coefficients for 16×16 blocks of the frame in Fig. 6.

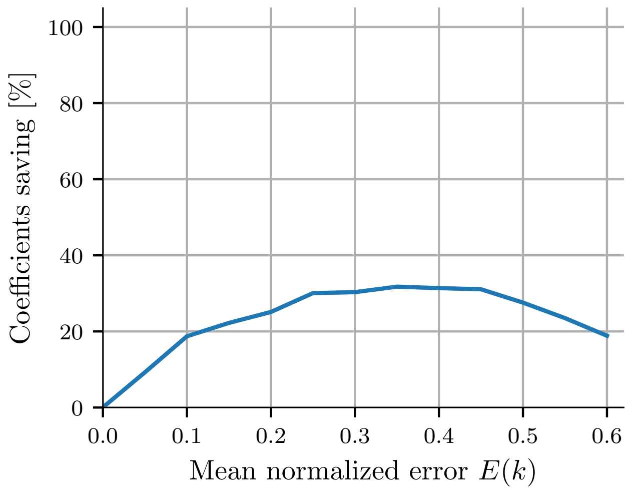


Fig. 8: Percentage of saving in the number of coefficients using DirLap compared to 2D-DCT for prescribed error levels in Fig. 7. For instance, 2D-DCT requires about 25% more coefficients than DirLap for $E(k) = 0.2$.

of about a factor of 4 in running time, both for the forward and inverse transforms.

VI. Evaluation

A. *Effectiveness of the Directional Transform Representation*

In this section the effectiveness of the representation of the directional transform is demonstrated and compared to 2D-DCT without using the complete HEVC framework. This is only meant to provide some intuition, as coding gains need to be measured inside a coding scheme.

For this demonstration we used 16×16 residual blocks of the image after intra-prediction, shown in Fig. 6, for which the intra-predictor is angular (neither DC nor planar). We applied both DirLap, 2D-DCT, and the Karhunen-Loeve Transform (KLT) to the residual blocks. The DirLap transform is in the direction of the intra-prediction. The KLT was trained on the blocks of the residual image itself. To measure the residual error after the addition of each transform coefficient, we ordered the 2D-DCT coefficients in diagonal up-right scan order, DirLap's coefficients in the order of eigenvalues of the directional system matrices, and KLT's coefficients in the order of eigenvalues of the empirical covariance matrix of the image blocks. Denote by $\mathbf{r}_j(k)$ the residual error at pixel j of some image block after the inverse transform with the first $k{=}1, 2, \ldots, 16^2$ coefficients from the coefficients vector $\mathbf{c}$. We can measure the effectiveness of each representation using a normalized energy error:

$$E(k) = 1 - \frac{\sum_{j=1}^{N^2} \mathbf{r}_j^2(k)}{\sum_{j=1}^{N^2} \mathbf{r}_j^2(N^2)} = 1 - \frac{\sum_{i=1}^{k} \mathbf{c}_i^2}{\sum_{i=1}^{N^2} \mathbf{c}_i^2} \quad . \tag{30}$$

Fig. 7 illustrates the rate of reduction of $E(k)$ as a function of k for the frame shown in Fig. 6. The wiggles in the 2D-DCT graph are a result of the diagonal scan, that walks between coefficients of horizontally-oriented and vertically-oriented basis functions. The residual image has some directional bias that did not average out with the averaging of the blocks.

Clearly, DirLap requires less coefficients than 2D-DCT for the same representation error, as quantified in Fig. 8. DirLap is actually more efficient than KLT in the left portion of Fig. 7. This is possible because DirLap uses multiple transforms, one for each prediction direction, in contrast to one global KLT in this comparison.

B. Coding results

We compared the coding performance of our implementation to the KTA-2.0 reference implementation on the Common Test Conditions (CTC) sequences [25]. The results are summerized in Table III. The standard Bjøntegaard Delta-Rate (BD-Rate) metric [32] is used for comparison. The DirLap results in 0.8%, 0.7%, 0.4% and 0.4% average coding gain on top of the KTA-2.0 reference, for All Intra (AI), Random Access (RA), Low Delay B (LDB), and Low Delay P (LDP) video profiles. Interestingly, for class F (screen content) the gain due to the introduction of the directional transforms is higher, between 0.8% and 1.4%.

Relative to the reference code, DirLap's average encoding and decoding times are 128% and 110% for AI, 107% and 105% for RA, 106% and 102% for LDB, and 109% and 108% for LDP. At the encoder's side, most of the increase in running time may be attributed to the RDOQ process, that each one of the transform candidates undergoes before selecting the optimal transform.

We measured the percentage of DirLap running time out of the total encoding time. DirLap's forward and inverse transforms are comparable to the most time consuming EMT transform (DST7). For example, in AI encoding of one frame of the sequence PartyScene using QP 22, DirLap spends 0.9% of the encoder's time in the forward transform, and 0.9% in the inverse transform, while DST7 takes 0.7% and 1.1% for forward and inverse transforms, respectively (here the summation is over all transform sizes, e.g. the 0.7% consists of 0.3%, 0.3% and 0.1% for DST7 of block sizes 16, 8 and 4).

TABLE III: Coding performance of DirLap over HM-16.6-KTA-2.0 reference for AI, RA, LDB and LDP on the standard CTC sequences. Empty cells are not required by the CTC standard. Negative numbers indicates coding gain, and Y is the channel of interest. Video dimensions are: Class A: 2560×1600, Class B: 1920×1080, Class C: 832×480, Class D: 416×240, Class E: 1280×720.

Compression profile	Videos class	Y	U	V
All Intra Main	Class A	-0.5%	-0.4%	-0.5%
	Class B	-0.8%	-0.4%	-0.4%
	Class C	-1.1%	-0.4%	-0.5%
	Class D	-0.7%	-0.2%	-0.1%
	Class E	-1.0%	-0.2%	-0.1%
	Overall mean	**-0.8%**	-0.3%	-0.4%
	Class F	-1.4%	-0.7%	-0.8%
	Encoding Time[%]		128%	
	Decoding Time[%]		110%	
Random Access Main	Class A	-0.4%	0.3%	0.4%
	Class B	-0.7%	0.1%	-0.2%
	Class C	-0.9%	-0.3%	-0.4%
	Class D	-0.6%	0.2%	-0.1%
	Class E			
	Overall mean	**-0.7%**	0.1%	-0.1%
	Class F	-1.4%	-0.7%	-0.6%
	Encoding Time[%]		107%	
	Decoding Time[%]		105%	
Low delay P Main	Class A			
	Class B	-0.4%	0.1%	0.0%
	Class C	-0.4%	0.1%	0.1%
	Class D	-0.2%	-1.0%	-0.1%
	Class E	-0.6%	-0.5%	-0.8%
	Overall mean	**-0.4%**	-0.3%	-0.2%
	Class F	-0.9%	-0.6%	-0.1%
	Encoding Time[%]		109%	
	Decoding Time[%]		108%	
Low delay B Main	Class A			
	Class B	-0.4%	0.0%	0.0%
	Class C	-0.4%	-0.7%	-0.2%
	Class D	-0.2%	-0.2%	-0.5%
	Class E	-0.6%	-1.0%	0.1%
	Overall mean	**-0.4%**	-0.4%	-0.1%
	Class F	-0.8%	-0.4%	0.0%
	Encoding Time[%]		106%	
	Decoding Time[%]		102%	

VII. Conclusion

This work introduces new directional transforms which are orthogonal bases for square blocks. The directional transforms are applicable for efficient representation of image data with directional patterns, as may be found in the residual of angular intra-prediction.

Historically, the 1D-DCT was derived as the eigenvectors of two different matrices. The first approach, by Ahmed, Natarajan and Rao [1] [2], approximates the KLT for stationary Markovian signals whose covariance matrix models exponential decaying covariance with the distance between points in 1D. It may be possible to derive 2D directional bases by modeling the covariance matrix of directional patterns, though we don't expect a clean analytical solution as the 1D-DCT.

The second approach by Strang [3] showed that 1D-DCT vectors are the eigenvectors of a second difference matrix that applies the discrete Laplacian operator. We followed this approach to obtain the directional Laplacian basis from the eigenvectors of the directional Laplacian operator. This operator is an analog of the 1D Laplacian (or second derivative operator) in a rotated system of coordinates [26]. We provided a discretization of the directional Laplacian operator and examined the resulting eigenvectors.

Although we focused on the 2D case, the scheme could be generalized to higher dimensions, e.g. to deal with volumetric data, by discretization of (13) and eigen-decomposition of the resulting system matrix. Depending on the application, one could also consider the Laplacian over a linear subspace, e.g. a slanted plane instead of a 1D direction, by summing a subset from the diagonal elements of the rotated Hessian (12). However, a higher dimensional implementation would be computationally heavy due to larger bases.

In our implementation, both the memory and running time of a naive implementation of the directional Laplacian transform were significantly reduced. Memory was reduced using symmetries in the transform directions, and symmetry or anti-symmetry in the eigenvectors. Running time was reduced using symmetry or anti-symmetry of the eigenvectors, as well as SIMD instructions.

We implemented the directional Laplacian transform as an additional transform inside KTA-2.0, that already includes multiple transforms (EMT [15]) and secondary transforms (ROT [16]) extensions to HEVC. We measured the coding gain on the CTC sequences [25]. For the all-intra class, an average BD-gain of 0.8% was achieved on top of KTA-2.0. To put our gains into perspective, according to Sullivan ([33] page 10) most tools in JEM1.0 contribute less than 1%. The exploding demand for video streaming makes every 1% of coding gain in video compression economically significant.

References

[1] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE transactions on Computers*, vol. 100, no. 1, pp. 90–93, 1974.

[2] K. R. Rao and P. Yip, *Discrete cosine transform: algorithms, advantages, applications*. Academic press, 1990.

[3] G. Strang, "The discrete cosine transform," *SIAM review*, vol. 41, no. 1, pp. 135–147, 1999.

[4] J. Lainema, F. Bossen, W.-J. Han, J. Min, and K. Ugur, "Intra coding of the hevc standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1792–1801, 2012.

[5] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[6] M. Karczewicz, J. Chen, W. Chien, X. Li, A. Said, L. Zhang, and X. Zhao, "Study of coding efficiency improvements beyond hevc," *MPEG doc. m37102*, vol. 10, 2015.

[7] J. Xu, B. Zeng, and F. Wu, "An overview of directional transforms in image coding," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*. IEEE, 2010, pp. 3036–3039.

[8] B. Zeng and J. Fu, "Directional discrete cosine transformsa new framework for image coding," *IEEE transactions on circuits and systems for video technology*, vol. 18, no. 3, pp. 305–313, 2008.

[9] R. A. Cohen, S. Klomp, A. Vetro, and H. Sun, "Direction-adaptive transforms for coding prediction residuals," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*. IEEE, 2010, pp. 185–188.

[10] H. Xu, J. Xu, and F. Wu, "Lifting-based directional dct-like transform for image coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 10, pp. 1325–1335, 2007.

[11] J. Xu, F. Wu, J. Liang, and W. Zhang, "Directional lapped transforms for image coding," *IEEE Transactions on Image Processing*, vol. 19, no. 1, pp. 85–97, 2010.

[12] Y. Ye and M. Karczewicz, "Improved h. 264 intra coding based on bi-directional intra prediction, directional transform, and adaptive coefficient scanning," in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*. IEEE, 2008, pp. 2116–2119.

[13] C. Lan, J. Xu, and F. Wu, "Enhancement of hevc using signal dependent transform (sdt)," *VCEG-AZ08*, 2015.

[14] A. Saxena and F. Fernandes, "Ce7: Mode-dependent dct/dst for intra prediction in video coding," *commitee input document JCTVC-D033, ISO/IEC/ITU-T Joint Collaborative Team on Video Coding, Daegu, Korea (January 2011)*, 2011.

[15] J. Chen, Y. Chen, M. Karczewicz, X. Li, H. Liu, L. Zhang, and X. Zhao, "Coding tools investigation for next generation video coding based on hevc," in *Applications of Digital Image Processing XXXVIII*, vol. 9599. International Society for Optics and Photonics, 2015, p. 95991B.

[16] E. Alshina, A. Alshin, J. Min, K. Choi, A. Saxena, and M. Budagavi, "Known tools performance investigation for next generation video coding," *ITU-T SG16/Q6 Doc. VCEG-AZ05*, 2015.

[17] X. Zhao, J. Chen, M. Karczewicz, A. Said, and V. Seregin, "Joint separable and non-separable transforms for next-generation video coding," *IEEE Transactions on Image Processing*, vol. 27, no. 5, pp. 2514–2525, 2018.

[18] H. Chen and B. Zeng, "New transforms tightly bounded by dct and klt," *IEEE Signal Processing Letters*, vol. 19, no. 6, pp. 344–347, 2012.

[19] H. Chen, S. Zhu, and B. Zeng, "Design of non-separable transforms for directional 2-d sources," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*. IEEE, 2011, pp. 3697–3700.

[20] I. W. Selesnick and O. G. Guleryuz, "A diagonally-oriented dct-like 2d block transform," in *SPIE 8138 (Wavelets and Sparsity XIV)*. International Society for Optics and Photonics, 2011.

[21] G. Fracastoro, S. M. Fosson, and E. Magli, "Steerable discrete cosine transform," *IEEE Transactions on Image Processing*, vol. 26, no. 1, pp. 303–314, 2017.

[22] Z. Gu, W. Lin, B.-S. Lee, and C. T. Lau, "Rotated orthogonal transform (rot) for motion-compensation residual coding," *IEEE Transactions on Image Processing*, vol. 21, no. 12, pp. 4770–4781, 2012.

[23] I. Dvir, A. Allouche, D. Drezner, A. Ecker, D. Irony, N. Peterfreund, H. Yang, and J. Zhou, "Trapezoidal block split using orthogonal c2 transforms for hevc video coding," in *European Signal Processing Conference (EUSIPCO)*. IEEE, 2017, pp. 1016–1020.

[24] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045–1057, 2010.

[25] F. Bossen, "Common test conditions and software reference configurations," Joint Collaborative Team on video Coding (JCT-VC), JCTVC-L1100, Geneva, 2013.

[26] M. Broadhead, "A directional laplacian with application to anisotropic convection-diffusion filtering," in *73rd EAGE Conference and Exhibition incorporating SPE EUROPEC*, 2011.

[27] B. Jahne, *Digital image processing*. Springer Berlin, 2005.

[28] W. A. Strauss, *Partial Differential Equations: An Introduction*. Wiley, 2007.

[29] J. Sole, R. Joshi, N. Nguyen, T. Ji, M. Karczewicz, G. Clare, F. Henry, and A. Duenas, "Transform coefficient coding in hevc," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1765–1777, 2012.

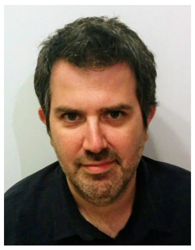
[30] https://hevc.hhi.fraunhofer.de/trac/jem/browser/tags/HM-16.6-KTA-2.0/.

[31] M. Budagavi, A. Fuldseth, G. Bjøntegaard, V. Sze, and M. Sadafale, "Core transform design in the high efficiency video coding (hevc) standard," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 6, pp. 1029–1041, 2013.

[32] G. Bjøntegaard, "Calculation of average psnr differences between rd-curves," *VCEG-M33*, 2001.

[33] G. Sullivan, "Video coding: Recent developments for hevc and future trends," https://www.cs.brandeis.edu/~dcc/Programs/Program2016KeynoteSlides-Sullivan.pdf, 2016.

Itsik Dvir is with Toga Networks, a Huawei company, Tel-Aviv Research Center, since 2013, where he initiates and leads the research in projects for VR 360 video and beyond HEVC video coding as a chief architect and researcher, respectively. His previous positions include initiating and leading R&D of real-time algorithms and architecture in various fields: video analytic for surveillance and retail products at Mango DSP / Mate Intelligent Video, during 2008-2013; video and image processing for digital still and video cameras at Zoran Microelectronics, during 2003-2008; biomedical signals for heart disease and sleep disorder diagnostic products at Itamar Medical, during 1999-2003; and statistical signal processing using antenna arrays for wireless and cellular communication, at RAFAEL, Haifa, during 1994-1999. Itsik received his B.Sc., M.Sc. and Ph.D. all in electrical engineering from the Technion - Israel Institute of Technology, in 1988, 1990 and 1994, respectively. There, he was a research fellow in 1994 and an adjunct lecturer. He has 8 US granted patents, 8 patent applications, and over 18 publications in scientific journals and international conferences.



Dror Irony received the B.Sc. in mathematics and computer science (1996), followed by M.Sc. (2000) and Ph.D. (2005) in computer science from Tel-Aviv University. Since 2005, he has held research positions at Orbotech (2005-2013), Huawei (2013-2016), Intel (2016-2017) and currently at Apple (from 2017). His research interests focus on computer vision, machine learning and deep learning, as well as image processing and computational linear algebra.

David Drezner received the M.Sc. degree from the Department of Electrical Engineering, Tel-Aviv University, Israel, in 2006. From 2003 to 2009, he was an algorithm and firmware team leader at Broadcom, developing advanced video pre-processing and compression algorithms. From 2009 to 2013, he was a senior algorithm engineer at Imagine Communications, developing advanced transcoders and post-processing video quality enhancements algorithms. From 2014 to 2017, he worked as a video compression researcher at Huawei. He is currently with the start-up company TetaVi as a chief science officer, working on computer vision, AI, 3D video reconstruction and compression applications.

Ady Ecker received the B.Sc. degree in mathematics and computer science from Tel-Aviv University in 1996, the M.Sc. degree in computer science from the Weizmann Institute of Science, Israel, in 2003, and the Ph.D. degree in computer science from the University of Toronto, in 2010. His research focused on optimization methods for geometric problems in computer vision, such as image alignment and 3D reconstruction. From 2010 to 2014, he was with Given Imaging (Medtronic), working on capsule endoscopy. From 2015 to 2016, he was with Toga Networks (Huawei), working on video compression. Since 2017, he is with Inuitive, working on 3D reconstruction from an active stereo system.

Amiram Allouche received the B.Sc. and M.Sc. degrees from the Department of Electrical Engineering, Technion, Israel, in 1998 and 2005, respectively. From 1998, Amiram held various R&D positions at Elbit Systems, Surf Communication Solutions and Imagine Communications as a developer and applied researcher. Since 2014, he is with Huawei TRC (Toga Networks) as a researcher and senior architect. His research interests include signal and image processing, video processing and delivery, computer vision and AI.

Natan Peterfreund is a senior research scientist at Amazon, working on computer vision and machine learning. He received his Ph.D. degree in electrical engineering from the Technion - Israel Institute of Technology, in 1994. Prior to joining Amazon, he led advanced research at Huawei's research lab and was a co-founder of Playcast Media Systems, which developed a cloud gaming service. His research interests include computer vision, machine learning and video compression.