■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

•Laplacian eigenfunctions: Each of the 16 possible functions $\widetilde{ff}$ has coordinate values that are sums of multiples of $\frac{1}{2}$ and $\sqrt{2}/2$. More specifically, if c is one of the 4 coordinate values of a function $\widetilde{ff}$, then c can only have a value that satisfies this condition:

$$c = \frac{1}{2}(I + J\sqrt{2}), \text{ where}$$
$$I \in \{-1, 0, 1, 2\} \text{ and}$$
$$J \in \{-1, 0, 1\}.$$

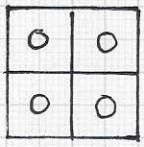
General form of $\widetilde{ff}$'s coordinates: $\frac{1}{2}(I+J\sqrt{2})$; table of all allowed possibilities:

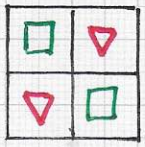
I	J	c^*	S
-1	-1	$-1-\sqrt{2}$	
-1	0	-1	
-1	1	$-1+\sqrt{2}$	
0	-1	$-\sqrt{2}$	
0	0	0	
0	1	$\sqrt{2}$	
1	-1	$1-\sqrt{2}$	
1	0	1	
1	1	$1+\sqrt{2}$	
2	-1	$2-\sqrt{2}$	
2	0	2	
2	1	$2+\sqrt{2}$	

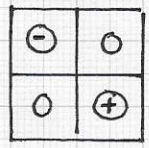
$c = c^*/2$

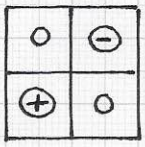
Symbol code for the 12 possible coefficient values. Each symbol 'S' represents a specific possible value of a coefficient c.

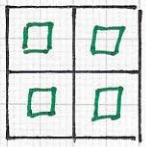
Considering the 12 possibilities for coordinate values, one can create a "visual encoding" of the 16 possible functions $\widetilde{ff}$ (left table):

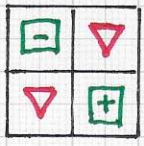

0000

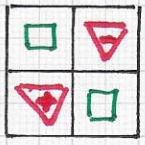

1000

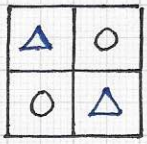

0100

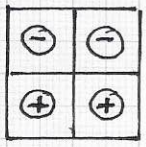

0010

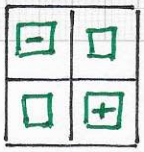

0001

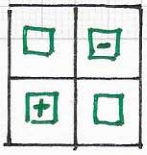

1100


1010


1001


0110


0101

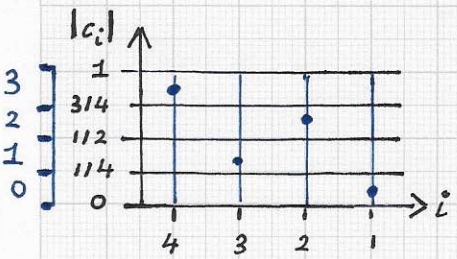

0011

Each binary code $b_1b_2b_3b_4$ defines the 4 binary coefficients defining $\widetilde{ff} = \sum_{i=1}^{4} b_i \cdot \widetilde{e}_i^n$.

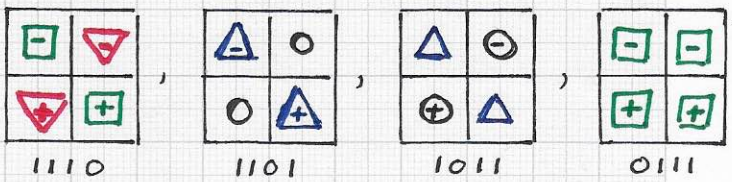
...

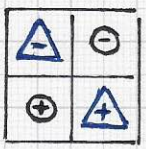
■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

•Laplacian eigenfunctions: ... Visual encoding of remaining functions $\tilde{ff}$:

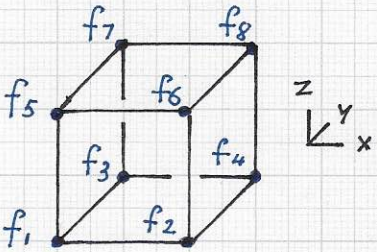


Four quantization levels make possible the use of $4^4 = 256$ codes already.

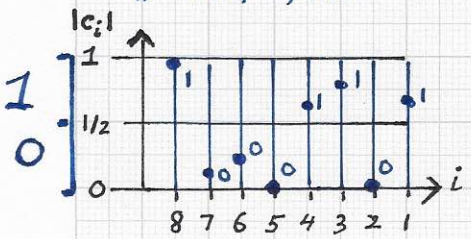




These 16 visual representations for the $16 = 2^4$ 4-bit quantizations / codes $(b_1 b_2 b_3 b_4)$ show that 2 quantization levels (0, 1) are capable of capturing distinct local behavior well. IF ONE WERE TO USE 4 QUATIZATION LEVELS (0, 1, 2, 3) TO QUANTIZE A GIVEN REAL-VALUED SET OF NORMALIZED COEFFICIENTS OF AN EXPANSION, ONE COULD ALREADY USE $4^4 = 256$ CODES FOR THE 2^2-PIXEL TEMPLATE. (See figure – left, top.)



$\tilde{ff} = (f_1, \ldots, f_8)^T$



$$\tilde{ff} = \sum_{i=1}^{8} c_i \tilde{e}_i^n$$

$\Rightarrow$ "8-bit code":

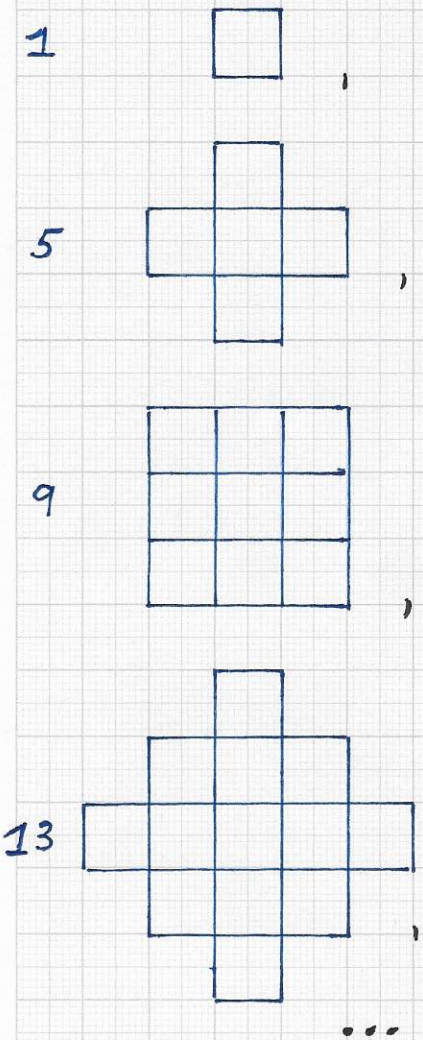
$$\tilde{ff} = \sum_{i=1}^{8} b_i \tilde{e}_i^n \,, \qquad b_i \in \{0, 1\} \,.$$

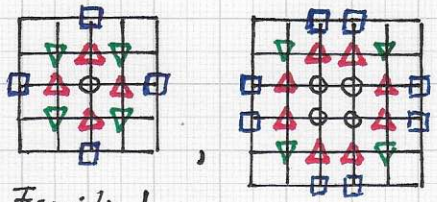
Two quantization levels make possible the use of $2^8 = 256$ codes in the 3D case, considering the local 2^3-voxel neighborhood.

Note: Considering the 3D case of a function $\tilde{ff}$ defined for a local $2^3 = 8$ voxel neighborhood, one can expand $\tilde{ff}$ via eigenfunctions as $\tilde{ff} = \sum_{i=1}^{8} c_i \tilde{e}_i^n$; when quantizing the real-valued coefficients with only two quantization levels (0, 1), one can represent $2^8 = 256$ local 8-bit codes characterizing $\tilde{ff}$'s behavior, i.e., codes of the form $(b_1, b_2, \ldots, b_8)$. (See left figure.)

■ OBJECT AND MATERIAL EIGENFUNCTIONS - Cont'd.

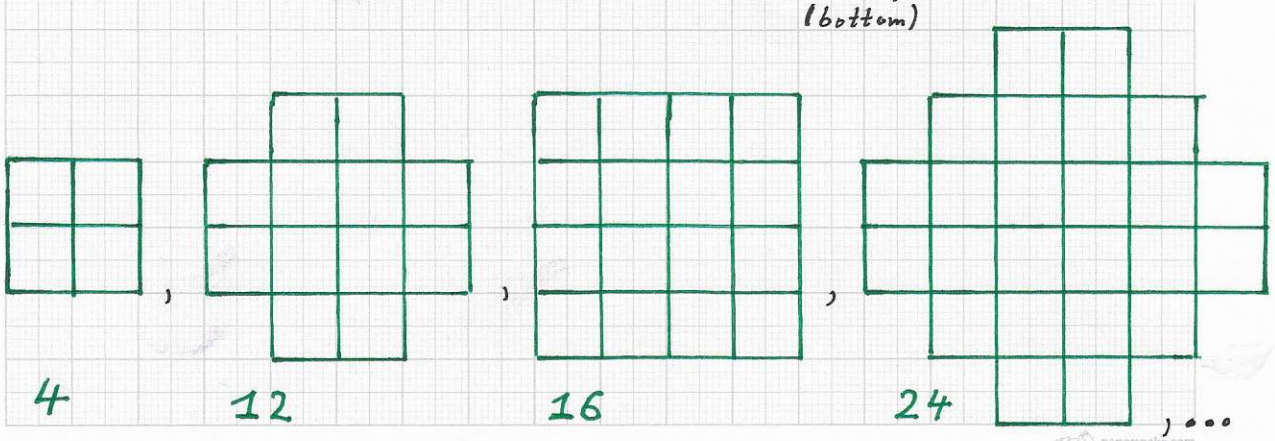
•Laplacian eigenfunctions: For computational efficiency, the choice of the local pixel-/voxel-template used to calculate a local spectrum of a given function ff (= a given segment) is important as it determines the numbers of inner products and matrix inversions that are needed to expand ff locally in a multi-frequency eigenbasis. We sketch low-resolution 2D pixel templates here. One family of template patterns leads to pixel neighborhoods of sizes 1, 5, 9, 13 etc. (left), while the other family produces sizes 4, 12, 16, 24 etc. (below). These pattern result by "concentric pixel addition": These patterns are "grown" by adding pixel layers iteratively (○, △, ▽, □).





Family 1 (left)

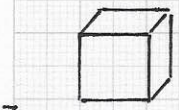
Family 2 (bottom)



Pixel neighborhoods one can use for local eigenfrequency analysis.

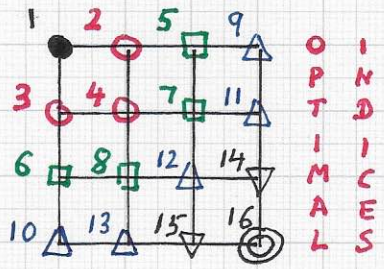
■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

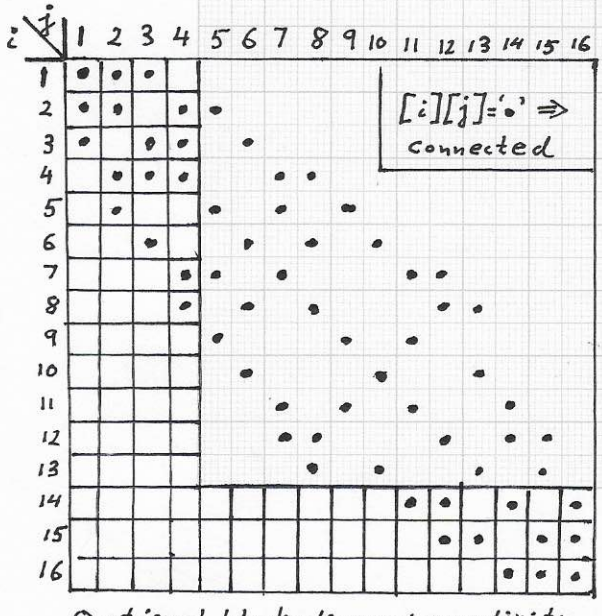
•Laplacian eigenfunctions: In the 3D voxel setting, one can construct these "concentrically grown voxel neighborhoods" analogously. The first family of patterns starts with one central voxel, and layers are added iteratively; the second family of patterns starts with an initial 2^3 voxel set, and layers are added around it one-by-one. The numbers of the resulting voxel templates grow rapidly for the two families of concentric patterns: the number sequence for the first family is 1, 7, 19, 27, ..., and the sequence for the second family is 8, 32, 56, 64, ... Of course, one must keep in mind this rapidly increasing number of voxels defining a template since it affects computational cost. In fact, it should be possible to obtain a "sufficiently good" eigenfrequency analysis and segment characterization with relatively small local voxel templates – "good enough for classification."



"Seeds" of one voxel and 2^3 voxel stencil used to construct two families of voxel neighborhoods for eigenanalysis.

Cuthill-McKee Indexing



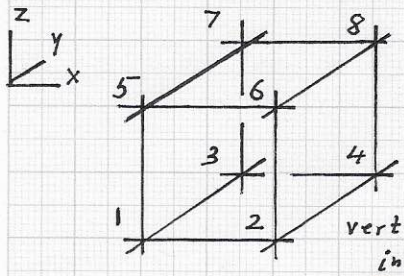


Optimal block-diagonal connectivity matrix.

The 2D pixel and 3D voxel neighborhoods/templates we consider limit the numbers of neighbors a pixel/voxel can have: a pixel has at most four and a voxel at most six neighbors, which is also reflected in the $M^{-1}K$ matrices. Thus, matrix bandwidth minimization (Cuthill-McKee algorithm) makes possible efficient computations.

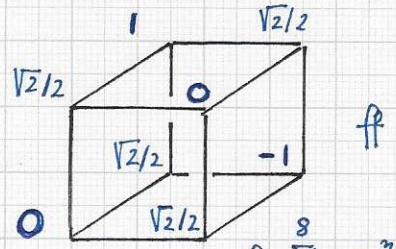
■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions:



NON-ORTHOGONAL eigen-basis for 2^3-voxel open template:

$$\mathbb{e}_1^n = (-1, 1, 1, -1, 1, -1, -1, 1)^T/2\sqrt{2},$$
$$\mathbb{e}_2^n = (1, 0, 0, -1, -1, 0, 0, 1)^T/2,$$
$$\mathbb{e}_3^n = (0, 1, 0, -1, -1, 0, 1, 0)^T/2,$$
$$\mathbb{e}_4^n = (0, 0, 1, -1, -1, 1, 0, 0)^T/2,$$
$$\mathbb{e}_5^n = (-1, 0, 0, 1, -1, 0, 0, 1)^T/2,$$
$$\mathbb{e}_6^n = (0, -1, 0, -1, 1, 0, 1, 0)^T/2,$$
$$\mathbb{e}_7^n = (0, 0, -1, -1, 1, 1, 0, 0)^T/2,$$
$$\mathbb{e}_8^n = (1, 1, 1, 1, 1, 1, 1, 1)^T/2\sqrt{2}.$$



$$\mathbb{ff} = \sum_{i=1}^{8} c_i \mathbb{e}_i^n$$

Pre-compute and store A^{-1}

⇒ compute $\mathbb{c}$ directly:

$$\begin{aligned}
c_1 &= b_1 \\
c_2 &= 3/2\, b_2 - 1/2\, b_3 - 1/2\, b_4 \\
c_3 &= -1/2\, b_2 + 3/2\, b_3 - 1/2\, b_4 \\
c_4 &= -1/2\, b_2 - 1/2\, b_3 + 3/2\, b_4 \\
c_5 &= 3/2\, b_5 + 1/2\, b_6 + 1/2\, b_7 \\
c_6 &= 1/2\, b_5 + 3/2\, b_6 - 1/2\, b_7 \\
c_7 &= 1/2\, b_5 - 1/2\, b_6 + 3/2\, b_7 \\
c_8 &= b_8
\end{aligned}$$

THUS: EFFICIENT PROCESSING IS POSSIBLE!

Linear systems of equations with block-diagonal matrices can be solved efficiently. As a simple example, we consider the expansion of the function $\mathbb{ff} = (0, \sqrt{2}/2, \sqrt{2}/2, -1, \sqrt{2}/2, 0, 1, \sqrt{2}/2)^T$, using vertex indexing as shown in the left figure. Expanding this function in the eigenbasis $\{\mathbb{e}_i^n\}_{i=1}^8$ consisting of linearly independent, normalized but NOT (fully) mutually orthogonal functions $\mathbb{e}_i^n$ leads to a "special" block-diagonal system to be solved:

$$\begin{bmatrix} 1 & & & & & & & \\ & 1 & 1/2 & 1/2 & & & & \\ & 1/2 & 1 & 1/2 & & & & \\ & 1/2 & 1/2 & 1 & & & & \\ & & & & 1 & -1/2 & -1/2 & \\ & & & & -1/2 & 1 & 1/2 & \\ & & & & -1/2 & 1/2 & 1 & \\ & & & & & & & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_8 \end{bmatrix} = \begin{bmatrix} 1 \\ 1/2 \\ 1 \\ 1/2 \\ -1/2 \\ 1 \\ 1/2 \\ 1 \end{bmatrix}$$

(See p. 1, 10/16/2021.)

$$\Leftrightarrow \quad \mathbb{A}\,\mathbb{c} = \mathbb{b}, \quad \mathbb{b} = (b_1, b_2, \ldots, b_8)^T.$$

Since $\mathbb{A}$ contains two "independent," non-overlapping 3-by-3 blocks, one can invert $\mathbb{A}$ efficiently and obtains $\mathbb{c}$: $\mathbb{c} = \mathbb{A}^{-1}\mathbb{b} \Leftrightarrow$

$$\mathbb{c} = \begin{bmatrix} 1 & & & & & & & \\ & 3/2 & -1/2 & -1/2 & & & & \\ & -1/2 & 3/2 & -1/2 & & & & \\ & -1/2 & -1/2 & 3/2 & & & & \\ & & & & 3/2 & 1/2 & 1/2 & \\ & & & & 1/2 & 3/2 & -1/2 & \\ & & & & 1/2 & -1/2 & 3/2 & \\ & & & & & & & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1/2 \\ 1 \\ 1/2 \\ -1/2 \\ 1 \\ 1/2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}.$$

Test: $\sum_{i=1}^{8} c_i \mathbb{e}_i^n = \mathbb{ff}$ ✓

~ BH