Nonseparable 2D Wavelet Filters

Susumu Sakakibara

School of Infromation Environment, Tokyo Denki University, Muzaigakuendai 2−1200, Inzai−city, Chiba−ken, Japan 270−1382
susumu@sie.dendai.ac.jp

In order to improve nonisotropic nature of the conventional tensor DWT on images [1], we have constructed new 2D nonseparable biorthogonal wavelets [3,4]. Filters are defined on the regular triangular Bravais lattice, which are constructed by lifting [2] polyphase components in a symmetrical arrangement. A nobel feature of our construction is that there are four independent sulattices, which corresponds to the even and odd indexed phases in the 1D polyphase representation. Correspondingly, there are one LP filter and three HP filters. We show some details of construction of the linear prediction triangular filters, and application to a simple image data. L^1 norm is evenly distributed over three detail components, implying that the isotropy of images are well respected.


```
In[1]:= Off[General::spell]
        Off[General::spell1]
        Needs["Graphics`Arrow`"]
        Needs["Graphics`Graphics`"]
        Needs["LinearAlgebra`MatrixManipulation`"]
```

■ Discrete Wavelet Transform (DWT)

A single decomposition step of DWT of 1D data with size 2^j is realized by the following Mallat matrix:

```
In[6]:= MallatMatrix[j_, s_] := Join[
          Table[h[Mod[k - 2 l, 2^j] - s], {l, 0, 2^(j-1) - 1}, {k, s, 2^j - 1 + s}],
          Table[g[Mod[k - 2 l, 2^j]], {l, 0, 2^(j-1) - 1}, {k, 0, 2^j - 1}]];
        MallatMatrix[j_] := MallatMatrix[j, 0]
```

which performs a filtering followed by downsampling with LP filter **h** and HP filter **g**.

For example, consider DWT of the data of size 2^3

```
In[7]:= simpleData1D = Table[a[k], {k, 0, 7}]

Out[7]= {a[0], a[1], a[2], a[3], a[4], a[5], a[6], a[7]}
```

with the Haar filters

```
In[8]:= Clear[h, g]; h[0] = h[1] = 1/Sqrt[2]; h[k_] := 0;
        g[n_] := 0; g[0] := -h[0]; g[1] := h[1]
```

The Mallat matrix in this size is

In[9]:=
```
MallatMatrix[3] // MatrixForm
```

Out[9]//MatrixForm=

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$$

By multiplying this Matrix to the data vector yiels the DWT decompotion

In[10]:=
```
MallatMatrix[3].simpleData1D
```

Out[10]=

$$\left\{\frac{a[0]}{\sqrt{2}}+\frac{a[1]}{\sqrt{2}}, \frac{a[2]}{\sqrt{2}}+\frac{a[3]}{\sqrt{2}}, \frac{a[4]}{\sqrt{2}}+\frac{a[5]}{\sqrt{2}}, \frac{a[6]}{\sqrt{2}}+\frac{a[7]}{\sqrt{2}}, -\frac{a[0]}{\sqrt{2}}+\frac{a[1]}{\sqrt{2}}, -\frac{a[2]}{\sqrt{2}}+\frac{a[3]}{\sqrt{2}}, -\frac{a[4]}{\sqrt{2}}+\frac{a[5]}{\sqrt{2}}, -\frac{a[6]}{\sqrt{2}}+\frac{a[7]}{\sqrt{2}}\right\}$$

consisting of coarse and detail components of size $2^2 = 4$, respectively. The inverse transform, reconstruction, is of course realized by the inverse matrix

In[11]:=
```
Expand[Inverse[MallatMatrix[3]].%] == simpleData1D
```

Out[11]=
```
True
```

For 2D image data

In[12]:=
```
simpleData2D = Table[b[n, m], {m, 0, 3}, {n, 0, 3}];
```

the DWT is usually carried out in the tensor product form,

In[13]:=
```
MallatMatrix[2].simpleData2D.
  Transpose[MallatMatrix[2]] // Expand
```

■ Lifting scheme

In the lifting scheme, the DWT decomposition is carried out by the following procedure. First, the data is decomposed into even and odd indexed samples, the step called the Lazy transform.

In[14]:=
```
{c, d} = Transpose[Partition[simpleData1D, 2]]
```

Out[14]=
```
{{a[0], a[2], a[4], a[6]}, {a[1], a[3], a[5], a[7]}}
```

The **d** component is subtracted by the predictor, which is assumed to be **c** in the Haar case.

In[15]:=
```
d = d - c
```

Out[15]=
```
{-a[0] + a[1], -a[2] + a[3], -a[4] + a[5], -a[6] + a[7]}
```

The **c** component is updated by adding **d/2** so that the total of **c** will be normalized to be a half of the original **a**.

In[16]:=
```
c = c + d / 2
```

Out[16]=
$$\left\{a[0] + \frac{1}{2}(-a[0] + a[1]),\ a[2] + \frac{1}{2}(-a[2] + a[3]),\right.$$
$$\left. a[4] + \frac{1}{2}(-a[4] + a[5]),\ a[6] + \frac{1}{2}(-a[6] + a[7])\right\}$$

Finally, the result is normalized as

In[17]:=
```
{c, d} = {√2 c, d / √2} // Expand
```

Out[17]=
$$\left\{\left\{\frac{a[0]}{\sqrt{2}} + \frac{a[1]}{\sqrt{2}},\ \frac{a[2]}{\sqrt{2}} + \frac{a[3]}{\sqrt{2}},\ \frac{a[4]}{\sqrt{2}} + \frac{a[5]}{\sqrt{2}},\ \frac{a[6]}{\sqrt{2}} + \frac{a[7]}{\sqrt{2}}\right\},\right.$$
$$\left.\left\{-\frac{a[0]}{\sqrt{2}} + \frac{a[1]}{\sqrt{2}},\ -\frac{a[2]}{\sqrt{2}} + \frac{a[3]}{\sqrt{2}},\ -\frac{a[4]}{\sqrt{2}} + \frac{a[5]}{\sqrt{2}},\ -\frac{a[6]}{\sqrt{2}} + \frac{a[7]}{\sqrt{2}}\right\}\right\}$$

so that the total energy is the same as the original signal **a**.

In[18]:=
```
Expand[Total[c^2 + d^2]] == Total[simpleData1D^2]
```

Out[18]=
```
True
```

■ Linear Prediction Filters

In the lifting scheme, the linear prediction filters are obtained by modifying the predicter and updator as

In[19]:=
```
{c, d} = Transpose[Partition[simpleData1D, 2]];
```

In[20]:=
```
d = d - (c + RotateLeft[c]) / 2
```

Out[20]=
$$\left\{a[1] + \frac{1}{2}(-a[0] - a[2]),\ a[3] + \frac{1}{2}(-a[2] - a[4]),\right.$$
$$\left. a[5] + \frac{1}{2}(-a[4] - a[6]),\ \frac{1}{2}(-a[0] - a[6]) + a[7]\right\}$$

In[21]:=
```
c = c + (d + RotateRight[d]) / 4
```

Out[21]=
$$\left\{a[0] + \frac{1}{4}\left(a[1] + \frac{1}{2}(-a[0] - a[2]) + \frac{1}{2}(-a[0] - a[6]) + a[7]\right),\right.$$
$$a[2] + \frac{1}{4}\left(a[1] + \frac{1}{2}(-a[0] - a[2]) + a[3] + \frac{1}{2}(-a[2] - a[4])\right),$$
$$a[4] + \frac{1}{4}\left(a[3] + \frac{1}{2}(-a[2] - a[4]) + a[5] + \frac{1}{2}(-a[4] - a[6])\right),$$
$$\left. a[6] + \frac{1}{4}\left(a[5] + \frac{1}{2}(-a[0] - a[6]) + \frac{1}{2}(-a[4] - a[6]) + a[7]\right)\right\}$$

```
In[22]:= {c, d} = {√2 c, d/√2} // Expand
```

$$\text{Out[22]}= \left\{\left\{\frac{3\,a[0]}{2\sqrt{2}}+\frac{a[1]}{2\sqrt{2}}-\frac{a[2]}{4\sqrt{2}}-\frac{a[6]}{4\sqrt{2}}+\frac{a[7]}{2\sqrt{2}},\right.\right.$$
$$-\frac{a[0]}{4\sqrt{2}}+\frac{a[1]}{2\sqrt{2}}+\frac{3\,a[2]}{2\sqrt{2}}+\frac{a[3]}{2\sqrt{2}}-\frac{a[4]}{4\sqrt{2}},$$
$$-\frac{a[2]}{4\sqrt{2}}+\frac{a[3]}{2\sqrt{2}}+\frac{3\,a[4]}{2\sqrt{2}}+\frac{a[5]}{2\sqrt{2}}-\frac{a[6]}{4\sqrt{2}},$$
$$\left.-\frac{a[0]}{4\sqrt{2}}-\frac{a[4]}{4\sqrt{2}}+\frac{a[5]}{2\sqrt{2}}+\frac{3\,a[6]}{2\sqrt{2}}+\frac{a[7]}{2\sqrt{2}}\right\},$$
$$\left\{-\frac{a[0]}{2\sqrt{2}}+\frac{a[1]}{\sqrt{2}}-\frac{a[2]}{2\sqrt{2}},-\frac{a[2]}{2\sqrt{2}}+\frac{a[3]}{\sqrt{2}}-\frac{a[4]}{2\sqrt{2}},\right.$$
$$\left.\left.-\frac{a[4]}{2\sqrt{2}}+\frac{a[5]}{\sqrt{2}}-\frac{a[6]}{2\sqrt{2}},-\frac{a[0]}{2\sqrt{2}}-\frac{a[6]}{2\sqrt{2}}+\frac{a[7]}{\sqrt{2}}\right\}\right\}$$

This is the decomposed coarse **c** component and detail **d** component, by the linear predition filter, usually called CDF(2,2).

By replacing **a[k]** by the following way, we obtain the transfer function of the LP filter $\hat{h}$ and the dual HP filter $\tilde{g}$.

```
In[23]:= ĥ[x_] = c[[2]] /. {a[n_] → Exp[-i (n - 2) x]}
```

$$\text{Out[23]}= -\frac{e^{-2ix}}{4\sqrt{2}}-\frac{e^{2ix}}{4\sqrt{2}}+\frac{e^{-ix}}{2\sqrt{2}}+\frac{e^{ix}}{2\sqrt{2}}+\frac{3}{2\sqrt{2}}$$

```
In[24]:= g̃[x_] = d[[2]] /. {a[n_] → Exp[-i (n - 2) x]}
```

$$\text{Out[24]}= -\frac{e^{-2ix}}{2\sqrt{2}}-\frac{1}{2\sqrt{2}}+\frac{e^{-ix}}{\sqrt{2}}$$

The filter coefficients may bo obtained similarly.

```
In[25]:= Clear[h, g]; h[k_] := 0;
         {h[-2], h[-1], h[0], h[1], h[2]} = List @@ c[[2]] /. a[k_] :↦ 1
```

$$\text{Out[25]}= \left\{-\frac{1}{4\sqrt{2}},\frac{1}{2\sqrt{2}},\frac{3}{2\sqrt{2}},\frac{1}{2\sqrt{2}},-\frac{1}{4\sqrt{2}}\right\}$$

```
In[26]:= g[k_] := 0; {g[0], g[1], g[2]} = List @@ d[[1]] /. a[k_] :↦ 1
```

$$\text{Out[26]}= \left\{-\frac{1}{2\sqrt{2}},\frac{1}{\sqrt{2}},-\frac{1}{2\sqrt{2}}\right\}$$

Thus, the Mallat matrix with these filters is

```
In[27]:= MallatMatrix[3, 2] // MatrixForm
```

$$\text{Out[27]//MatrixForm}= \begin{pmatrix}
\frac{3}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{4\sqrt{2}} & 0 & 0 & 0 & -\frac{1}{4\sqrt{2}} & \frac{1}{2\sqrt{2}} \\
-\frac{1}{4\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{3}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{4\sqrt{2}} & 0 & 0 & 0 \\
0 & 0 & -\frac{1}{4\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{3}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{4\sqrt{2}} & 0 \\
-\frac{1}{4\sqrt{2}} & 0 & 0 & 0 & -\frac{1}{4\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{3}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} \\
-\frac{1}{2\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{2\sqrt{2}} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -\frac{1}{2\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{2\sqrt{2}} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\frac{1}{2\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{2\sqrt{2}} & 0 \\
-\frac{1}{2\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2\sqrt{2}} & \frac{1}{\sqrt{2}}
\end{pmatrix}$$

■ Bravais Lattice

In order to construct our filters on the regular triangular lattice, let us define the primitive translation vectors

```
In[28]:= t[0] = {0, 0}; t[1] = {1, 0};
         t[2] = {-1 / 2, √3 / 2}; t[3] = -t[1] - t[2];
```

which generate the trianglar lattice called the Bravais lattice in two dimension. The 2 dimensional signal **c** is assigned on the lattice sites

```
In[29]:= Clear[c]; lattice = Table[c[n, m], {n, 4, -3, -1}, {m, -4, 3}];
         MatrixForm[lattice]
```

Out[29]//MatrixForm=

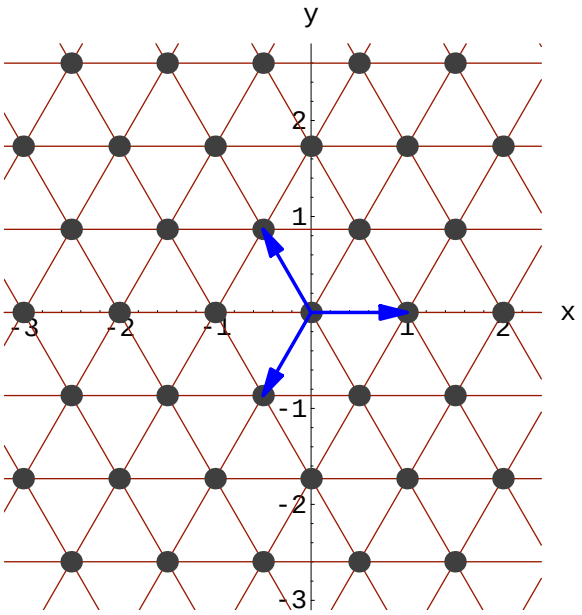
c[4, -4]	c[4, -3]	c[4, -2]	c[4, -1]	c[4, 0]	c[4, 1]	c[4, 2]	c[4, 3]
c[3, -4]	c[3, -3]	c[3, -2]	c[3, -1]	c[3, 0]	c[3, 1]	c[3, 2]	c[3, 3]
c[2, -4]	c[2, -3]	c[2, -2]	c[2, -1]	c[2, 0]	c[2, 1]	c[2, 2]	c[2, 3]
c[1, -4]	c[1, -3]	c[1, -2]	c[1, -1]	c[1, 0]	c[1, 1]	c[1, 2]	c[1, 3]
c[0, -4]	c[0, -3]	c[0, -2]	c[0, -1]	c[0, 0]	c[0, 1]	c[0, 2]	c[0, 3]
c[-1, -4]	c[-1, -3]	c[-1, -2]	c[-1, -1]	c[-1, 0]	c[-1, 1]	c[-1, 2]	c[-1, 3]
c[-2, -4]	c[-2, -3]	c[-2, -2]	c[-2, -1]	c[-2, 0]	c[-2, 1]	c[-2, 2]	c[-2, 3]
c[-3, -4]	c[-3, -3]	c[-3, -2]	c[-3, -1]	c[-3, 0]	c[-3, 1]	c[-3, 2]	c[-3, 3]

where, the value of the signal at the site **m t[1] + n t[2]** is denoted by **c[n, m]**. The lattice together with the primitive translation vectors are plotted as

```
In[30]:= plattice = lattice /. c[n_, m_] → Point[m t[1] + n t[2]];
```

```
In[31]:= pCell[n_, m_] := Line[{n t[1] + m t[2], (n + 1) t[1] + m t[2],
           (n + 1) t[1] + (m + 1) t[2], n t[1] + m t[2]}]
```

```
In[32]:= Show[
           Graphics[{RGBColor[0.6, 0.1, 0], Table[pCell[n, m], {n, -5, 3},
              {m, -4, 4}], {GrayLevel[.25], PointSize[0.04], plattice},
             {Thickness[0.0065], RGBColor[0, 0, 1], Arrow[t[0], t[1]],
              Arrow[t[0], t[2]], Arrow[t[0], t[3]]}},
            AspectRatio → Automatic, Axes → Automatic,
            PlotRange → {{-3.2, 2.4}, {-3.1, 2.8}},
            AxesLabel → {"x", "y"}]];
```



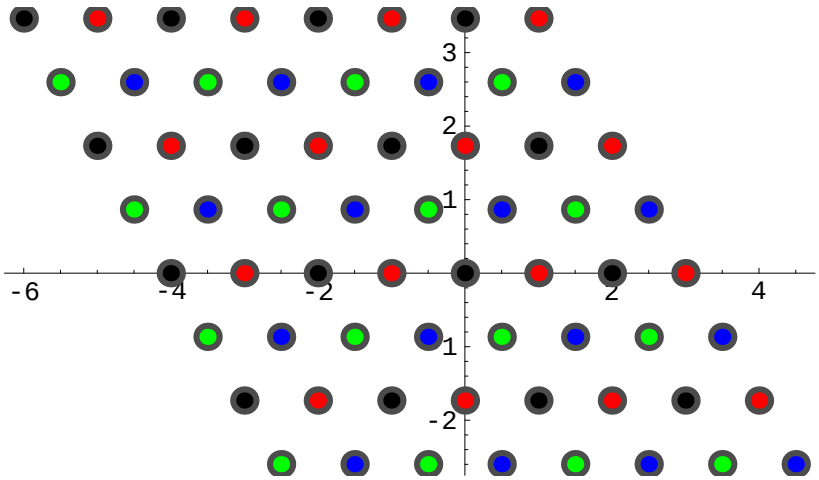
It is crucial to observe that the lattice sites may be devided into four independent sublattices, each of which is obtained by shifting zero or one unit of primitive translation vectors followed by scaling the original lattice by factor of two. To show this, we define the function that splits the lattice sites into the four sublattices.

In[33]:=
```
Lazy[mat_] := Module[{size}, size = Length[mat];
  {Take[mat, {1, size, 2}, {1, size, 2}],
   Take[mat, {1, size, 2}, {2, size, 2}],
   Take[RotateRight[mat], {1, size, 2}, {1, size, 2}],
   Take[RotateLeft[Map[RotateRight, mat]],
    {1, size, 2}, {1, size, 2}]}]
```

It is easier to understand the four sublattices by plotting them in different colors.

In[34]:=
```
Module[{lazyLattice}, lazyLattice = Lazy[plattice]; allSites =
   Graphics[{GrayLevel[.3], PointSize[0.035], plattice}];
  sites0 = Graphics[{PointSize[0.02], lazyLattice[[1]]}];
  sites1 = Graphics[
    {PointSize[0.02], RGBColor[1, 0, 0], lazyLattice[[2]]}];
  sites2 = Graphics[{PointSize[0.02], RGBColor[0, 1, 0],
     lazyLattice[[3]]}]; sites3 = Graphics[
    {PointSize[0.02], RGBColor[0, 0, 1], lazyLattice[[4]]}];];
```

In[35]:=
```
Show[{allSites, sites0 (* Gray *), sites1
   (* Red *), sites2(* Green *), sites3 (* Blue *)},
  AspectRatio → Automatic, Axes → Automatic];
```



The unshifted sublattice (black dots), the **t[1]** shifted sublattice (red dots), the **t[2]** shifted sublattice (green dots), the **t[3]** shifted sublattice (blue dots) add up to form the original lattice (gray circles). These sublattices correspond to the even and odd phases of 1D data vectors.

■ Triangular Filter

In order to extend the same steps as in the 1D linear prediction filters, let us define the functions that shift data in three directions.

In[36]:=
```
Shift[0][m_] := m;
Shift[1][m_] := Map[RotateRight, m];
Shift[-1][m_] := Map[RotateLeft, m];
Shift[2][m_] := RotateLeft[m];
Shift[-2][m_] := RotateRight[m];
Shift[3][m_] := RotateRight[Map[RotateLeft, m]];
Shift[-3][m_] := RotateLeft[Map[RotateRight, m]];
```

The triangular decomposition of the 2D data is carried out by the function

```
In[43]:= TriTwistDWT[data2D_] := Module[{c0, d1, d2, d3, td1, td2, td3},
           {c0, td1, td2, td3} = Lazy[data2D];
           d1 = -td1 + td2 + td3; d2 = td1 - td2 + td3;
           d3 = td1 + td2 - td3; d1 = d1 - (c0 + Shift[-1][c0]) / 2;
           d2 = d2 - (c0 + Shift[-2][c0]) / 2;
           d3 = d3 - (c0 + Shift[-3][c0]) / 2; c0 = c0 +
             (d1 + Shift[1][d1] + d2 + Shift[2][d2] + d3 + Shift[3][d3]) / 8;
           {2 c0, d1 / 2, d2 / 2, d3 / 2} // Expand]
```

which is essentially a straightforward generalization of the 1D DWT, except for the twist of three detail components in the third line of the above definition. Without the twist, we would get **TriStraightDWT**, but in this paper I would like to confine myself to the twisted version. As always, the inverse transform is given by tracing the same operations backward.

The data on the lattice is decomposed by the function **TriTwistDWT**.

```
In[44]:= dwt = TriTwistDWT[lattice];
```

which consists of four components, LP and three HP components. When the data **c[n, m]** are replaced by exponetials, the four components yields the transfer functions of LP and three HP filters. There are four LP transfer functions in the first element, which are all equivalent due to periodicity. By choosing an appropriate one, which turns out to be the (3,3) element here, they are

```
In[45]:= ĥ[x_, y_] =
           dwt[[1, 3, 3]] /. c[n_, m_] → Exp[-i m {x, y}.t[1] - i n {x, y}.t[2]]
```

$$\text{Out[45]}= \frac{5}{4} + \frac{e^{-ix}}{4} - \frac{e^{ix}}{4} - \frac{1}{8}e^{-2ix} - \frac{1}{8}e^{2ix} + \frac{1}{4}e^{-i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)} - \frac{1}{4}e^{i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)} - \frac{1}{8}e^{-2i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)} - \frac{1}{8}e^{2i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)} - \frac{1}{4}e^{-ix-i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)} + \frac{1}{4}e^{2ix-i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)} + \frac{1}{4}e^{ix+i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)} + \frac{1}{4}e^{3ix+i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)} - \frac{1}{8}e^{-2ix-2i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)} + \frac{1}{4}e^{-3ix-2i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)} + \frac{1}{4}e^{-ix+2i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)} - \frac{1}{8}e^{2ix+2i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)} + \frac{1}{4}e^{-2ix-3i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)} + \frac{1}{4}e^{ix+3i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)}$$

```
In[46]:= ĝ[1][x_, y_] =
           dwt[[2, 3, 3]] /. c[n_, m_] → Exp[-i m {x, y}.t[1] - i n {x, y}.t[2]]
```

$$\text{Out[46]}= -\frac{1}{4} - \frac{e^{-ix}}{2} - \frac{1}{4}e^{-2ix} + \frac{1}{2}e^{-i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)} + \frac{1}{2}e^{ix+i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)}$$

```
In[47]:= ĝ[2][x_, y_] =
           dwt[[3, 3, 3]] /. c[n_, m_] → Exp[-i m {x, y}.t[1] - i n {x, y}.t[2]]
```

$$\text{Out[47]}= -\frac{1}{4} + \frac{e^{-ix}}{2} - \frac{1}{2}e^{-i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)} - \frac{1}{4}e^{-2i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)} + \frac{1}{2}e^{ix+i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)}$$

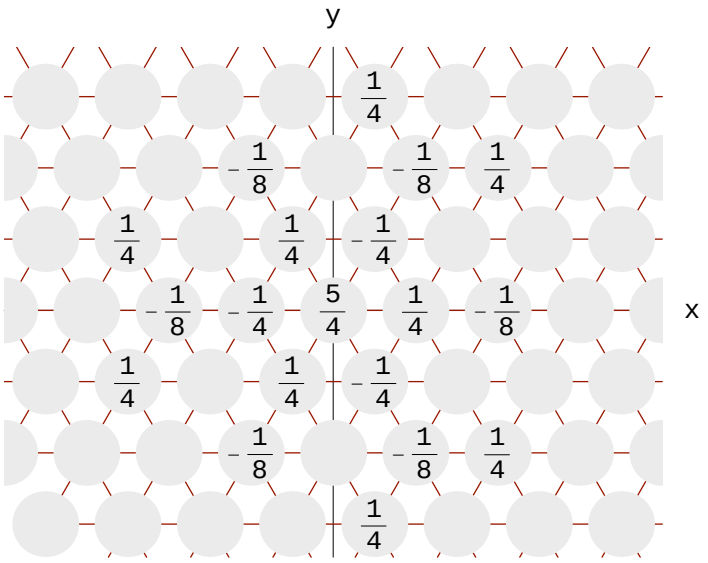
```
In[48]:= ĝ[3][x_, y_] =
           dwt[[4, 3, 3]] /. c[n_, m_] → Exp[-i m {x, y}.t[1] - i n {x, y}.t[2]]
```

$$\text{Out[48]}= -\frac{1}{4} + \frac{e^{-ix}}{2} + \frac{1}{2}e^{-i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)} - \frac{1}{2}e^{ix+i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)} - \frac{1}{4}e^{2ix+2i\left(\frac{\sqrt{3}y}{2}-\frac{x}{2}\right)}$$

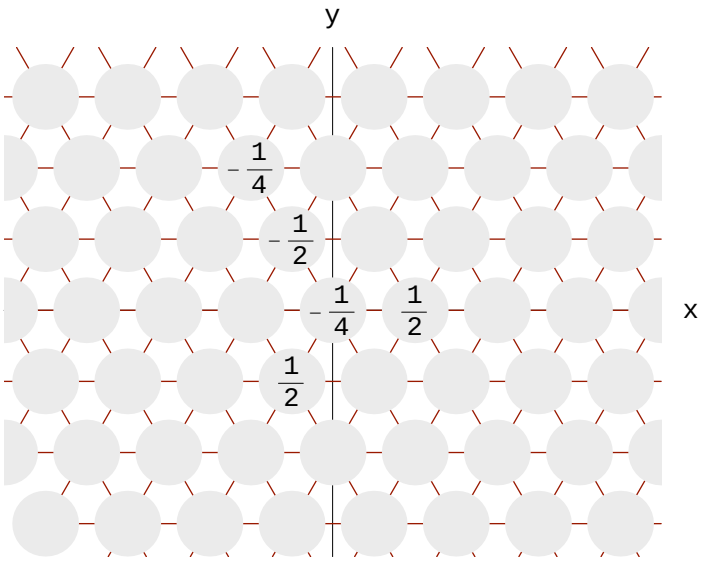
The array of filter coefficients on the lattice can by visulized as

In[49]:=
```
FilterCoeffs[expr_] := Module[{expl, coef},
  expl = Apply[List, expr]; coef = expl /. {x → 0, y → 0};
  expl = Expand[PowerExpand[i Log[expl / coef]]];
  coef = Transpose[{coef,
     Transpose[{expl /. {x → 1, y → 0}, expl /. {x → 0, y → 1}}]}];
  Map[Text[#[[1]], #[[2]]] &, coef]]
```

In[50]:=
```
Show[Graphics[{{GrayLevel[.92], PointSize[0.1],
    Table[{{RGBColor[.6, .1, 0], pCell[n, m]},
      Point[n t[1] + m t[2]]}, {n, -5, 6}, {m, -4, 3}]},
   FilterCoeffs[ĥ[x, y]]}, AspectRatio → Automatic,
  Axes → Automatic, Ticks → None,
  PlotRange → {{-4, 4}, {-3, 3.2}},
  AxesLabel → {"x", "y"}]];
```

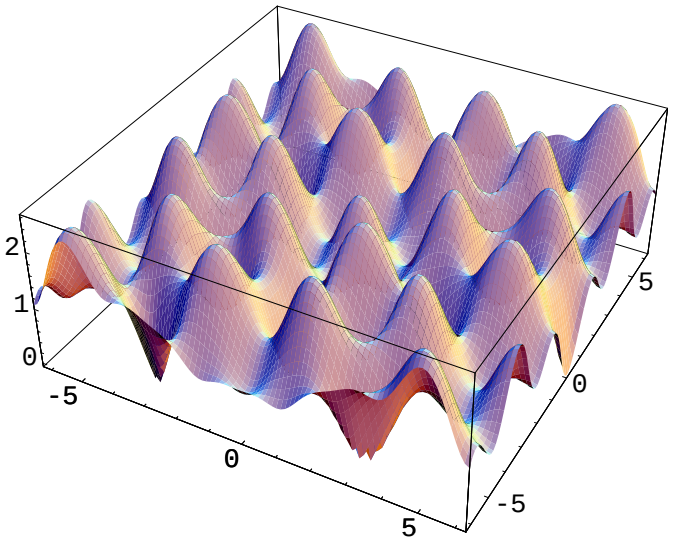


```
In[51]:= Show[Graphics[{{GrayLevel[.92], PointSize[0.1],
           Table[{{RGBColor[.6, .1, 0], pCell[n, m]},
             Point[n t[1] + m t[2]]}, {n, -5, 6}, {m, -4, 3}]},
         FilterCoeffs[ĝ[2][x, y]]}, AspectRatio → Automatic,
        Axes → Automatic, Ticks → None,
        PlotRange → {{-4, 4}, {-3, 3.2}},
        AxesLabel → {"x", "y"}]];
```



The other HP components are 120 degrees rotation of the last one. The 3D plot of $\hat{h}$ is

```
In[52]:= Plot3D[Abs[ĥ[x, y]], {x, -2 π, 2 π},
           {y, -2 π, 2 π}, PlotPoints → 100, Mesh → False];
```



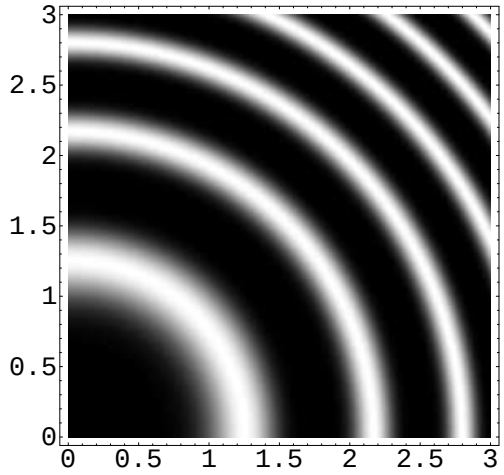
■ Image Processing

In[53]:=
```
ShowPyramid[dwtimg_] :=
 Module[{grf}, grf = Show[DensityGraphics[#, Mesh → False,
       Frame → None, DisplayFunction → Identity]] & /@ dwtimg;
  Show[GraphicsArray[Partition[grf, 2], GraphicsSpacing → 0,
    DisplayFunction → $DisplayFunction]]]
```

In[54]:=
```
L1Norm[dwtimg_] := Module[{en},
  en = Map[Total[Flatten[Abs[#]]] &, dwtimg] // N; en / Total[en]]
```

Let us consider a sample image

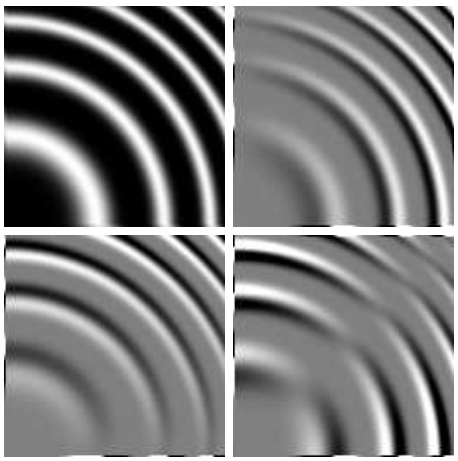
In[55]:=
```
gCircle = With[{rg = 3}, DensityPlot[Sin[x^2 + y^2]^6,
    {x, 0, rg}, {y, 0, rg}, PlotPoints → 256, Mesh → False]];
```



In[56]:=
```
dwtTriCircle = TriTwistDWT[gCircle[[1]]];
```

The coarse (upper left) and three deail images are

```
In[57]:= ShowPyramid[dwtTriCircle]; eCircle = L1Norm[dwtTriCircle]
```

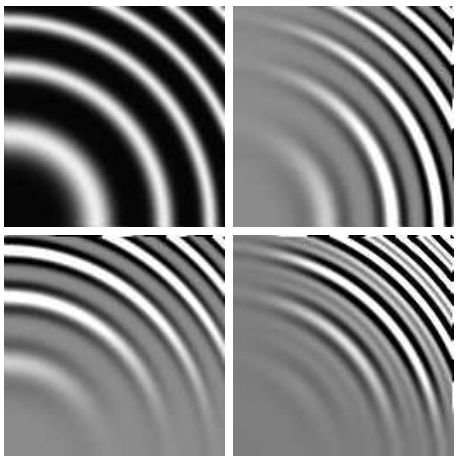


```
Out[57]= {0.867152, 0.0496598, 0.0492033, 0.0339844}
```

where the last line gives the corresponding normalized L^1 norm. Note that the norm of three detail components are roughly the same, implying that the decomposition is isotropic. On the other hand, in the conventional tensor form of DWT, the LL, LH, HL, HH components bocome

```
In[58]:= T = MallatMatrix[8, 2];
         dwtTensor =
           First[Map[{SubMatrix[#, {1, 1}, {128, 128}], SubMatrix[#,
                {1, 129}, {128, 128}], SubMatrix[#, {129, 1}, {128, 128}],
              SubMatrix[#, {129, 129}, {128, 128}]} &,
             {T.gCircle[[1]].Transpose[T]}]];
```

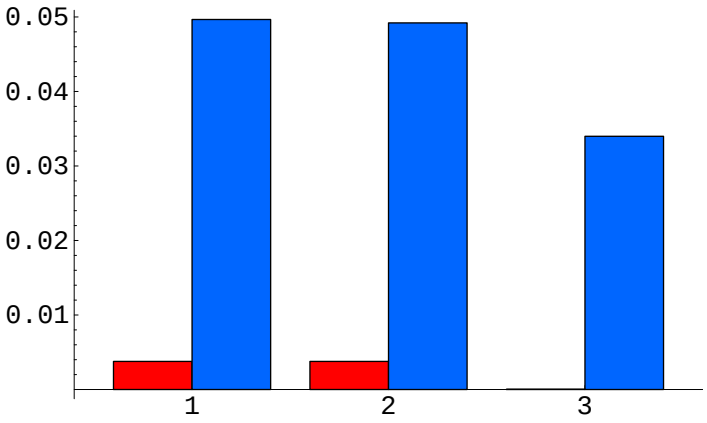
```
In[60]:= ShowPyramid[dwtTensor]; eCircleTensor = L1Norm[dwtTensor]
```



```
Out[60]= {0.992424, 0.00377142, 0.00377142, 0.0000334818}
```

We see that the HH components carries much less energy compared with LH and HL components. It becomes clear when the normalized L^1 norm of the three detail components are compared. The blue bar corresponds to our triangular DWT.

```
In[61]:= BarChart[Rest[eCircleTensor], Rest[eCircle]];
```



■ Conclusion

It is shown that filters defined on the Bravais lattice appears very promising for image processing, but their implementation involves rather delicate data manipulations, such as the distinction between **RotateRight**, **RotateLeft**. In the above method, the same definition is used in sereral places, in producing expressions and plots, thereby consistency will be checked. Such a proceedure is certainly one of the strongest features of *Mathematica.*

As mentioned in the text, there are straight (untwisted) version of the DWT, and many other generalizations to filters other than linear prediction filter. The inverse transform in any case is realized by taking the reverse steps of lifting, and hence it is easy to implement. With such tools at hand, we can investigate image processing of various types of images, including textures, bounded variation, etc. It would also be interesting to visualize the corresponding scaling function and wavelets, and study their regularity. Such investigations are currently under study, which are not duscussed here, due to space limitations.

■ References

[1] S. Mallat, *A Wavelet Tour of Signal Processing*, 2nd ed., Academic Press, 2001

[2] W. Sweldens, *The lifting scheme: a custom−design construction of biorthogonal wavelets*, Applied and Computational Harmonic Analysis, vol.3(2), pp.186−200, 1996

[3] S. Sakakibara, *Aspects of Wavelet Analysis*, Proceedings of SICE 2005, paper no. MP1−03−5, 2005

[4] S. Sakakibara and O. Vasilyev, *Image Processing with Triangular Biorthogonal Wavelets*, paper submitted to eusipco2006