Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

PNA

Probability, Networks and Algorithms

Probability, Networks and Algorithms

A toolbox for the lifting scheme on quincunx grids (LISQ)

P.M. de Zeeuw

REPORT PNA-R0224 DECEMBER 31, 2002

CWI is the National Research Institute for Mathematics and Computer Science. It is sponsored by the Netherlands Organization for Scientific Research (NWO).
CWI is a founding member of ERCIM, the European Research Consortium for Informatics and Mathematics.

CWI's research has a theme-oriented structure and is grouped into four clusters. Listed below are the names of the clusters and in parentheses their acronyms.

Probability, Networks and Algorithms (PNA)

Software Engineering (SEN)

Modelling, Analysis and Simulation (MAS)

Information Systems (INS)

P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

ISSN 1386-3711

A Toolbox for the Lifting Scheme on Quincunx Grids (LISQ)

P.M. de Zeeuw
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
Paul.de.Zeeuw@cwi.nl

ABSTRACT
A collection of functions written in MATLAB® is presented. The functions include second generation wavelet decomposition and reconstruction tools for images as well as functions for the computation of moments. The wavelet schemes rely on the lifting scheme of Sweldens and use the splitting of rectangular grids into quincunx grids, also known as red-black ordering. The prediction filters include the Neville filters as well as a nonlinear $maxmin$ filter. Custom-made filters can be used too. The various functions are described and examples are given. The toolbox is provided with appliances for the visualization of data on quincunx grids. The software can be downloaded from a website and is publicly available.

2000 Mathematics Subject Classification: 62H35, 65T60, 65Y15, 94A08
1998 ACM Computing Classification System: G.1.0, G.4
Keywords and Phrases: Discrete wavelets, lifting scheme, MATLAB®, moments, quincunx, red-black ordering, second generation wavelets, software, toolbox, wavelets.
Note: This work was carried out under project PNA4.2 "Image Representation and Analysis".

1. Introduction

The toolbox LISQ is built on the lifting scheme and uses quincunx downsampling of rectangular two-dimensional grids. An early example of quincunx downsampling and upsampling (in multigrid context) can be found in [2]. Early examples of quincunx downsampling in connection with 2-channel multidimensional filter banks can be found in [13, 14]. The lifting scheme has been invented by Sweldens [10, 11]. In [6, 15, 16, 17] the lifting scheme is used with quincunx downsampling to develop non-separable wavelets on a rectangular grid. An educational and introductory approach to the lifting scheme (in 1D) can be found in [5].

LISQ can do the following for you. This toolbox performs the wavelet decomposition of a 2D-signal (image) and corresponding reconstruction. The dimensions of the grid on which the image is defined need not be dyadic. Prediction (and update) filters can be chosen from predefined sets, but custom-made filters are possible too. Additionally, means for the computation of moments (on both rectangular and quincunx grids) are present. Visualization of data on quincunx grids is provided for. The toolbox does not require any other toolbox.

Please note that for convenience the option of inplace computations (as made possible by the lifting scheme) is not (always) used. Still, the toolbox is efficient with cpu-time because of persistent vectorization of computations. Applications can be in the region of denoising, image fusion and image retrieval. The toolbox may be especially helpful for research purposes. The contents are readily available and can be downloaded from the web (see Section 5).

2. Background

In this section we sketch the background of the toolbox and its components. Firstly, we recapitulate on quincunx grids and the lifting scheme. We proceed from a simple red-black version of the lifting

scheme to ones involving Neville filters and nonlinear filters. Secondly, we discuss the (numerical) computation of moments on both rectangular and quincunx grids.

2.1 Quincunx Grids & The Lifting Scheme

Quincunx grids Let us consider an image as a two-dimensional signal. We subdivide the lattice on which the signal has been defined into two sets on quincunx grids as indicated in Figure 1. This

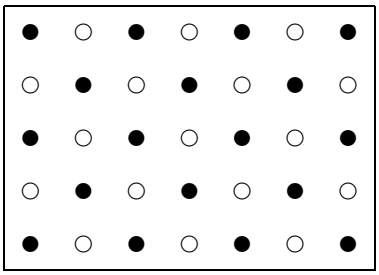

Figure 1: A rectangular grid composed of two quincunx grids.

division is also called "checkerboard" or "red-black" division. An early example of this division can be found in [2], where a quincunx grid is called "intermediate checkered grid". There, in multigrid context, it serves the purpose of a nonstandard coarsening (downsampling) of discrete differential operators. Also in multigrid context, the ordering is used in the so-called red-black relaxation because of its decoupling properties in the case of standard five-point discretization.

The lifting scheme As extensive literature exists on this topic, (e.g. [1, 5, 6, 10, 11, 12, 15, 16, 17]) we confine ourselves to a basic recapitulation. We consider a n-dimensional signal $s_j \in \mathcal{S}(S_j)$ as a function s_j: $S_j \to \mathbb{R}$ where $S \subset \mathbb{Z}^n$, $n \in \mathbb{N}$. We transform s_{j-1} into a coarser, approximating, signal s_{j-1} and a detail signal d_{j-1} such that $S_{j-1} \subsetneq S_j$ (downsampling) and $S_j = S_{j-1} \cup D_{j-1}$, $S_{j-1} \cap D_{j-1} = \emptyset$ (splitting). The lifting scheme can be described by the following algorithm:

Decomposition

$$s_{j-1} := s_j \downarrow_{S_{j-1}}; \tag{2.1a}$$
$$d_{j-1} := s_j \downarrow_{D_{j-1}}; \tag{2.1b}$$
$$d_{j-1} := d_{j-1} - P(s_{j-1}); \quad \text{(substract prediction)} \tag{2.1c}$$
$$s_{j-1} := s_{j-1} + U(d_{j-1}); \quad \text{(update)} \tag{2.1d}$$

where

$$P: \ \mathcal{S}(S_{j-1}) \to \mathcal{S}(D_{j-1}) \tag{2.2a}$$
$$U: \ \mathcal{S}(D_{j-1}) \to \mathcal{S}(S_{j-1}) \tag{2.2b}$$

and $\downarrow_{S_{j-1}}$ denotes downsampling $\mathcal{S}(S_j) \to \mathcal{S}(S_{j-1})$. The computations can be done in-place [12]. The inverse scheme reads:

Reconstruction

$$s_{j-1} := s_{j-1} - U(d_{j-1}); \tag{2.3a}$$
$$d_{j-1} := d_{j-1} + P(s_{j-1}); \tag{2.3b}$$
$$s_j := s_{j-1} \uparrow^{S_j} + d_{j-1} \uparrow^{S_j}; \tag{2.3c}$$

where $\uparrow^{S_j}$ denotes upsampling $\mathcal{S}(S_{j-1}) \to \mathcal{S}(S_j)$.

order N	V_1	V_2	V_3	V_4	V_5	V_6	V_7
2	$1/4$	0	0	0	0	0	0
4	$10/32$	$-1/32$	0	0	0	0	0
6	$87/2^8$	$-27/2^9$	2^{-8}	$3/2^9$	0	0	0
8	$5825/2^{14}$	$-2235/2^{15}$	$625/2^{16}$	$425/2^{15}$	$-75/2^{16}$	$9/2^{16}$	$-5/2^{12}$

Table 1: Quincunx Neville filter coefficients

The red-black transform A first example of the lifting scheme involving quincunx grids is the red-black wavelet transform by Uytterhoeven and Bultheel [15, 16, 17]. A rectangular grid is split into two quincunx grids as in Figure 1. The pixels on the red spots ($\circ$) are used to predict the samples on the black spots ($\bullet$), while updating of the red spots is performed by using the detailed data on the black spots. The second order prediction and update filters are given by

$$(\mathcal{P}x)(i,j) = [x(i-1,j)+x(i,j-1)+x(i+1,j)+x(i,j+1)]/4,\ i \bmod 2 \neq j \bmod 2, \quad (2.4a)$$
$$(\mathcal{U}x)(i,j) = [x(i-1,j)+x(i,j-1)+x(i+1,j)+x(i,j+1)]/8,\ i \bmod 2 = j \bmod 2. \quad (2.4b)$$

Neville filters and the lifting scheme In general a prediction filter $\mathcal{P}$ for the quincunx grid can be written as

$$(\mathcal{P}x)(i,j) = \sum_{(n,m)\in S_{\tilde{N}}} a_{\tilde{N}}(n,m)\, x(i+n,j+m),\ i \bmod 2 \neq j \bmod 2, \quad (2.5)$$

with $S_{\tilde{N}}$ a subset of $\{(n,m) \in \mathbb{Z}^2 \mid (n+m) \bmod 2 = 1\}$ and $a_{\tilde{N}}(s)$, $s \in S_{\tilde{N}}$, a set of coefficients in $\mathbb{R}$. In this case a general formula for $\mathcal{U}$ reads

$$(\mathcal{U}x)(i,j) = \sum_{(n,m)\in S_N} a_N(n,m)\, x(i+n,j+m)/2,\ i \bmod 2 = j \bmod 2, \quad (2.6)$$

with S_N depending on the number of required primal vanishing moments N. For several elements in S_N the coefficients $a_N(s)$ attain the same values. Therefore we take these elements together in subsets of S_N, i.e.,

$$\begin{aligned}
V_1 &= \{(+1,0),(0,+1),(-1,0),(0,-1)\},\\
V_2 &= \{(+1,+2),(-1,+2),(-2,+1),(-2,-1),(-1,-2),(+1,-2),(+2,-1),(+2,+1)\},\\
V_3 &= \{(+3,0),(0,+3),(-3,0),(0,-3)\},\\
V_4 &= \{(+2,+3),(-2,+3),(-3,+2),(-3,-2),(-2,-3),(+2,-3),(+3,-2),(+3,+2)\},\\
V_5 &= \{(+1,+4),(-1,+4),(-4,+1),(-4,-1),(-1,-4),(+1,-4),(+4,-1),(+4,+1)\},\\
V_6 &= \{(+5,0),(0,+5),(-5,0),(0,-5)\},\\
V_7 &= \{(+3,+4),(-3,+4),(-4,+3),(-4,-3),(-3,-4),(+3,-4),(+4,-3),(+4,+3)\}.
\end{aligned} \quad (2.7)$$

Table 1 indicates the values of all $a_N(s)$, $s \in V_k$, for different values of N (2 through 8) when using quincunx Neville filters, see [6], which are the filters we use in our approach. The case $N = 2$ reduces to the red-black transform described before. The case $N = 8$ implies that $S_8 = V_1 + \cdots + V_7$ and so 44 taps filters are used as prediction/update filters. For an illustration of the Neville filter of order 4 see Figure 2. Here the numbers 1, 2 correspond to the values of the filter coefficients as given in V_1 and V_2 respectively at that position. The left-hand filter can be used to transform a signal defined on a quincunx grid into a signal defined on a rectangular grid, the right-hand filter is the 45 degrees rotated version of the left-hand filter and can be used to transform a signal from a rectangular grid towards a quincunx grid. We observe that the quincunx lattice yields a non separable 2D-wavelet transform, symmetric in both horizontal and vertical direction.

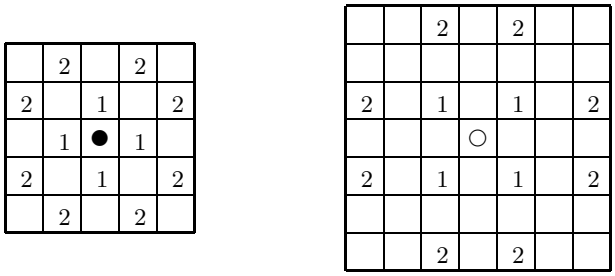

Figure 2: Neville filter of order 4: rectangular (left) and quincunx (right)

Maxmin: a nonlinear transform The above described prediction and update filters are all linear ones. The toolbox includes a scheme which is nonlinear. It is very like the red-black transform, but for the prediction and update filters. When the scheme proceeds from a rectangular grid towards a quincunx grid, the prediction filter is defined by assigning the maximum value at the neighbouring gridpoints, instead of computing the average as done in (2.4). When the scheme proceeds from a quincunx grid towards a rectangular grid the prediction filter is defined by assigning the minimum value at the neighbouring gridpoints. The update filters are similarly defined. The resulting nonlinear scheme is better in preserving in edges and local maxima and minima than the linear schemes which tends to blur the approximate images at coarser grids. For a more detailed description of the various operators involved we refer to [3] and to the annotated contents of the MATLAB® procedure `QLiftDec2MaxMin`.

Computational aspects Though a geometrical interpretation (see Figure 1) of a quincunx grid is straightforward, its computational representation looks awkward. However, the four-colour division, illustrated by Figure 3, shows that each of the quincunx grids can be seen as the union of two colours (compare Figures 3 and 1) each of which corresponds to ordinary rectangular grids. This proves to

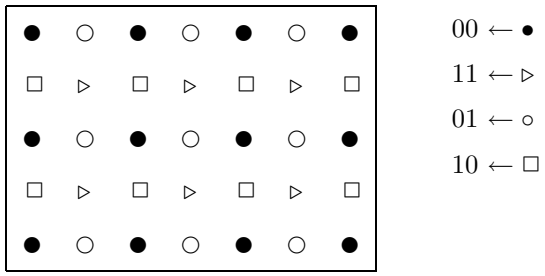

Figure 3: Four-colour division of a grid.

be an elegant and efficient approach for dealing with quincunx grids.

2.2 Moments & Invariants

We use invariants based on moments of the coefficients up to third order. Traditionally, these features have been widely used in pattern recognition applications to recognize the geometrical shapes of different objects. For the construction we follow Hu [4]. We take a density distribution function f. The $(p+q)$th order central moment $\mu_{pq}(f)$ of f is given by

$$\mu_{pq}(f) = \int_{\mathbb{R}} \int_{\mathbb{R}} (x - x_c)^p (y - y_c)^q f(x, y)\, d(x - x_c)\, d(y - y_c), \tag{2.8}$$

with the center of mass

$$x_c = \frac{\int\int_{\mathbb{R}\,\mathbb{R}} x f(x,y)\,dx\,dy}{\int\int_{\mathbb{R}\,\mathbb{R}} f(x,y)\,dx\,dy} \quad \text{and} \quad y_c = \frac{\int\int_{\mathbb{R}\,\mathbb{R}} y f(x,y)\,dx\,dy}{\int\int_{\mathbb{R}\,\mathbb{R}} f(x,y)\,dx\,dy}. \tag{2.9}$$

Computing the centers of mass x_c' and y_c' of $g(x,y) = f(x-a, y-b)$ yields $x_c' = x_c - a$, $y_c' = y_c - b$. Combining this with (2.8) shows that $\mu_{pq}(f) = \mu_{pq}(g)$, i.e., the central moments are translation invariant. We elaborate on some numerical aspects with respect to quincunx grids and the numerical computation of moments.

Numerical computation of moments We elaborate briefly on the numerical computation of moments. Figures 4 and 5 elucidate the computation of moments on a rectangular and quincunx grid respectively.

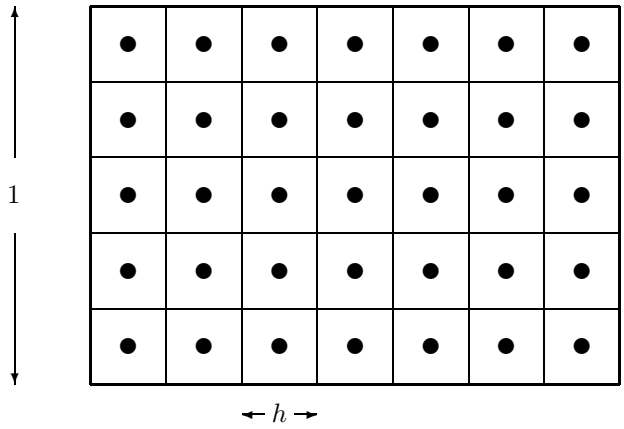

Figure 4: Supports at rectangular grid.

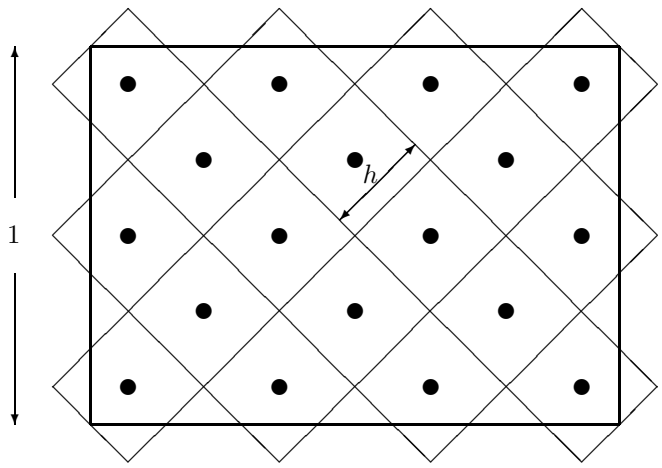

Figure 5: Supports at quincunx grid.

To each point • a value has been associated: this can be a grayvalue at a pixel, a wavelet coefficient, etc. Using these values we construct an interpolating function based on piecewise constant

approximation. For the rectangular grid the piecewise constant basisfunctions have their support on the squares, for the quincunx grid the piecewise constant basisfunctions have their support on the diamonds (rotated squares). A practical assumption is that the shortest side of the rectangular domain has size 1. As the supports are square, the dimension of the longest side of the domain also follows at once. Though the above interpolating function is not in the Schwartz class $S(\mathbb{R}^2)$, it has compact support, is measurable and can be integrated. Hereby we can now perform the integration in (2.8)– (2.9) numerically and thereby compute the moments. We note the following. When a function $f \in S(\mathbb{R}^2)$ is approximated piecewise constantly with an increasing number of gridpoints (pixels) then, roughly speaking, the moments remain invariant.

The homogeneity condition The following orthogonal (and translational) invariants (the last one skew, distinguishes mirror images) have been derived by Hu [4]

$$
\begin{aligned}
I_1 &= \mu_{20} + \mu_{02}, && \text{(2.10a)}\\
I_2 &= (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2, && \text{(2.10b)}\\
I_3 &= (\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2, && \text{(2.10c)}\\
I_4 &= (\mu_{30} + \mu_{12})^2 + (\mu_{21} + \mu_{03})^2, && \text{(2.10d)}\\
I_5 &= (\mu_{30} - 3\mu_{12})(\mu_{30} + \mu_{12})((\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2) + \\
&\quad (3\mu_{21} - \mu_{03})(\mu_{21} + \mu_{03})(3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2), && \text{(2.10e)}\\
I_6 &= (\mu_{20} - \mu_{02})((\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2) + 4\mu_{11}(\mu_{30} + \mu_{12})(\mu_{21} + \mu_{03}), && \text{(2.10f)}\\
I_7 &= (3\mu_{21} - \mu_{03})(\mu_{30} + \mu_{12})((\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2) - \\
&\quad (\mu_{30} - 3\mu_{12})(\mu_{21} + \mu_{03})(3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2). && \text{(2.10g)}
\end{aligned}
$$

Naively, we might compose the following feature vector $I \in \mathbb{R}^7$:

$$I \equiv (I_1 \quad I_2 \quad I_3 \quad I_4 \quad I_5 \quad I_6 \quad I_7)^T. \tag{2.11}$$

Without scaling of the elements the computation of the Euclidean norm of the difference between such feature vectors leads to arbitrary results. Moreover, the various elements appear to operate in different orders of magnitude. We therefore proceed as follows. Firstly, we observe that

$$\mu_{pq}(\lambda f) = \lambda \mu_{pq}(f), \text{ for all } \lambda \neq 0. \tag{2.12}$$

The homogeneity condition means that we demand a homogeneous change in the elements of a feature vector if the density distribution f is multiplied by the said scalar λ. Obviously, this condition is not satisfied by (2.10)–(2.11). We introduce the following operator:

$$\mathcal{R}_p(u) = \text{ sign}(u)|u|^{1/p}, \text{ for } p \in \mathbb{N} \text{ and } u \in \mathbb{R}. \tag{2.13}$$

When applied to an invariant I_k it produces again an invariant. It can easily be verified that the feature vector

$$\tilde{I} \quad = \quad (I_1 \quad \mathcal{R}_2(I_2) \quad \mathcal{R}_2(I_3) \quad \mathcal{R}_2(I_4) \quad \mathcal{R}_4(I_5) \quad \mathcal{R}_3(I_6) \quad \mathcal{R}_4(I_7))^T, \tag{2.14}$$

does satisfy the homogeneity condition.

3. Manual

The manual includes high-level procedures, the ones that are likely to be called by a user. With respect to the computation of moments we list: `mupq`, `Q0011mupq`, `momentsupto3`. With respect to the algorithm of the quincunx lifting scheme we list: `printshop`, `QLiftDec2`, `QLiftRec2`, `QLmaxlev`, `retrieveQ1001`, `retrieveR`, `rota1001fill`, `whatcoef2QL`. The (many) subsidiary procedures are not listed. The function and description of the subsidiary procedures can be found by using the `help` command in MATLAB®. By the above procedures one can understand the involved example of Section 4.

momentsupto3

Purpose Computes the mass, all central moments upto third order and normalized Hu's invariants of a two-dimensional signal (image, matrix) on a rectangular grid.

Syntax `[mass, mus, orthos] = momentsupto3(F)`

Description `momentsupto3` is a utility for the computation of moments and invariants.

Input `F` is a two-dimensional signal (image, matrix).

Output `mass` of type double is the mass of gridfunction `F`.

Output `mus`: a vector of length 7 containing all 2nd & 3rd order central moments (see (2.8)):

$$(\mu_{20} \quad \mu_{11} \quad \mu_{02} \quad \mu_{30} \quad \mu_{21} \quad \mu_{12} \quad \mu_{03})^T$$

Output `orthos`: a vector of length 7 containing (weighted) orthogonal invariants based on above central moments, see (2.14).

Algorithm The weighing of the elements of the vectors is performed by application of the so-called "homogeneity condition" [9, §5.2], see Section 2.2.

See Also `HUinvariants`, `Q0011momentsupto3`, `Q1001momentsupto3`.

Example

```
load zenithgray; Orig = zenithgray; clear zenithgray;
%
[mass, mus, orthos] = momentsupto3(Orig)

mass =

  369.4901


mus =

   36.4342    0.1725  108.4892   -0.4755    0.6690    0.2279    3.0241


orthos =

  144.9234   72.0559    1.5422    3.7014    2.4960    9.9251   -2.8705
```

mupq

Purpose Computes the central moments with prescribed order of a two-dimensional signal (image, matrix) on a rectangular grid.

Syntax

```
val = mupq(F, p, q)

val = mupq(F, p, q, Center)
```

Description `mupq` is a utility for the computation of moments and invariants. It computes the central moments μ_{pq} of order `p` in the x-direction and of order `q` in the y-direction and the signal `F` seen as a density distribution defined on a rectangular grid. Note that by the values $(p, q) = (1, 0)$ or $(p, q) = (0, 1)$ the central moment has to vanish.

`p` must be an integer, it is the order `p` in the x-direction.

`q` must be an integer, it is the order `q` in the y-direction.

`Center` is an optional argument, it is an array of 2 doubles that is supposed to be the center of mass. Without this argument the center of mass is computed for by this procedure. See also Hu [4].

Output `val` is the value of the central moment as computed.

Algorithm See Section 2.2.

See Also `m00`, `m10`, `m01`, `xcpowp`, `ycpowp`, `masscenter`, `Q1001mupq`.

Example

```
load zenithgray; Orig = zenithgray; clear zenithgray;
%
%---COMPUTE MASS-------------------------
mu00 = mupq(Orig, 0, 0)

mu00 =

  369.4901

%---NICE CHECK: OUTCOME SHOULD VANISH---
mu10 = mupq(Orig, 1, 0)

mu10 =

  -1.5131e-14

%---NICE CHECK: OUTCOME SHOULD VANISH---
mu01 = mupq(Orig, 0, 1)

mu01 =

  -9.1327e-14
```

Compare to the result of the example with procedure `Q0011mupq`.

printshop

Purpose `printshop` shows and files two-dimensional images.

Syntax

```
printshop(figthis, figtxt, f, dops, dotiff, mask, fmin, fmax)

printshop(figthis, figtxt, f, dops, dotiff, mask)
```

Description `printshop` is a utility for showing (and filing) two-dimensional images.

`figthis` must be a string of characters and is the name of the picture to be.

`figtxt` must be a string of characters and is the title of the picture to be.

`f` must be a two-dimensional array of doubles, it represents the image that has to be shown.

`dops` must be an integer, it steers the postscript imaging (and filing) of `f`. The possible values read:
0: perform no action whatsoever.
1: show the .epsc (colour) imagefile in Matlabs window.
2: as 1 but also file imagefile.
3: show the .eps (b & w) imagefile in Matlabs window.
4: as 3 but also file imagefile.

`dotiff` must be an integer, it steers the filing in .tif format of `f`. The possible values read:
0: perform no action whatsoever.
1: file the imagefile (.tif format) in current directory.
2: file the imagefile (.tif format) in the directory `/hosts/homepage/$USER/tmp/`. Obviously, one should adapt this to one's own environment.

`mask` must be a two-dimensional array of integers with values either 0 or 1, or it is an empty matrix. Its purpose is to mask f before imaging. If not empty, the dimensions of `mask` must be identical to the dimensions of `f`. The value 1 within the mask indicates to retain the original value of `f` in the image at the corresponding pixel, the value 0 indicates to replace the original value by a value greater then or equal to the maximum value of `f` (depending on the imaging this may then turn up as a white-coloured pixel). If the mask is empty, no masking is performed. The masking can be used as a tool to produce a desired background for a rotated image of a signal on a quincunx grid.

`fmin`, `fmax` are optional arguments. This way a minimum and a maximum of `f` (implying scaling) are enforced before imaging.

See Also `image`, `imagesc`, `imshow`.

Example

```
load zenithgray; Orig = zenithgray; clear zenithgray;
printshop("original","Nice watch", Orig, 3, 0, []);
```

See also the more involved example with `rota1001fill`.

QLiftDec2

Purpose Multi-level 2D composition by the lifting scheme involving quincunx grids.

Syntax `[C, S] = QLiftDec2(X, N, filtername)`

Description `QLiftDec2` performs the `N`-level decomposition of a two-dimensional signal (matrix, image) `X` by the lifting scheme using prediction and update filters that are indicated by `filtername`.

Outputs are the decomposition vector `C` and the corresponding bookkeeping matrix `S`. At odd levels of the decomposition the coefficients reside on quincunx-shaped grids, at even levels of the decomposition the coefficients reside on rectangular grids. The decomposition vector `C` includes all detail coefficients (at levels $<$ `N`) and the coefficients of the approximation (at level = `N`).

`X` must be a two-dimensional array, the dimensions need NOT be dyadic.

`N` must be a strictly positive and even integer (see `QLmaxlev`).

`filtername` must be a string from the set { `Neville2`, `Neville4`, `Neville6`, `Neville8`, `MaxMin` }. For a custom-made decomposition see `QLiftDec2Custom`.

Algorithm See Section 2.1.

See Also `QLiftRec2`, `QLmaxlev`.

Example

```
%---PARAMETERS-----------------------------------------------------------------
% How to execute, set parameters
N = 6;                    %  maximum level (even number) in lifting scheme
filtername = 'Neville4';
%
%---INSERT YOUR IMAGE HERE-----------------------------------------------------
if exist('imread','file') == 2
  Orig = double(imread('zenithgray.TIF','tiff'));
else
  load zenithgray; Orig = zenithgray; clear zenithgray;
end
%
sizeOrig = size(Orig)

sizeOrig =

   593   307

%
%---DECOMPOSITION--------------------------------------------------------------
[C,S] = QLiftDec2(Orig,N,filtername);
sizeC = size(C)

sizeC =
```

```
      182051           1

sizeS = size(S)

sizeS =

    10     6

S

S =

           1           4           1         296         154       45585
           1           3           1         297         153       91026
           2           0           1         296         153      136314
           3           4           1         148          77      147710
           3           3           1         149          77      159183
           4           0           1         148          77      170579
           5           4           1          74          39      173465
           5           3           1          75          38      176315
           6           0           1          74          38      179127
           6           0           0          75          39      182052
```

QLiftRec2

Purpose Multi-level 2D reconstruction by inverting the lifting scheme involving quincunx grids.

Syntax `X = QLiftRec2(C, S, filtername)`

Description `QLiftRec2` performs the reconstruction of a two-dimensional signal (matrix, image) `X` by inverting the lifting scheme using prediction and update filters that are indicated by `filtername`. The reconstruction involves the vector of coefficients `C` and the bookkeeping matrix `S`. The structure and dimensions of `C` and `S` are supposed to be consistent with how they are produced by `QLiftDec2`. `QLiftRec2` is the inverse function of `QLiftDec2`. `filtername` is supposed to be the same string as chosen for `QLiftDec2`.

Output `X` is a two-dimensional signal (image).

Algorithm See Section 2.1.

See Also `QLiftDec2`, `QLmaxlev`.

Example

```
%---PARAMETERS-----------------------------------------------------------------
% How to execute, set parameters
N = 6;        %  maximum level (even number) in lifting scheme
filtername = 'Neville4';
%
%---INSERT YOUR IMAGE HERE-----------------------------------------------------
if exist('imread','file') == 2
  Orig = double(imread('zenithgray.TIF','tiff'));
else
  load zenithgray; Orig = zenithgray; clear zenithgray;
end
%
sizeOrig = size(Orig)

sizeOrig =

   593   307

%---DECOMPOSITION--------------------------------------------------------------
[C,S] = QLiftDec2(Orig,N,filtername);
%
%---RECONSTRUCTION-------------------------------------------------------------
X = QLiftRec2(C,S,filtername);
%
%---THE 2-NORM OF THE DIFFERENCE BETWEEN THE ORIGINAL AND ITS RECONSTRUCTION--
  diff = X - Orig;
  twonormdiff = sum(sum(diff.*diff))^(1/2)

twonormdiff =

   2.5431e-12
```

QLmaxlev

Purpose Determines the maximum level to be used in the lifting scheme decomposition given the dimensions of a two-dimensional signal (image, matrix).

Syntax `lev = QLmaxlev(sizeX, filtername)`

Description `QLmaxlev` is a utility for the lifting scheme decomposition. It helps one to avoid silly values for the maximum level in the lifting scheme decomposition.

`lev` is the integer outcome.

`sizeX` is an integer row vector of dimension 2. Usually it will be the size of an image.

`filtername` must be a string from the set { `Neville2`, `Neville4`, `Neville6`, `Neville8`, `MaxMin` }.

Algorithm Basically, the algorithm determines how many times the grid can be coarsened until the size of the filter indicated by `filtername` exceeds the dimensions of the approximation.

See Also `QLiftDec2`.

Example

```
%---PARAMETERS-----------------------------------------------------------------
% How to execute, set parameters
N = 20;        %  maximum level (even number) in lifting scheme
filtername = 'Neville4';
%
%---INSERT YOUR IMAGE HERE-----------------------------------------------------
load zenithgray; Orig = zenithgray; clear zenithgray;
%
%---AVOID SILLY VALUES FOR THE NUMBER OF LEVELS--------------------------------
disp([' Number of scales asked for is ' num2str(N)]);
 Number of scales asked for is 20
sizeOrig = size(Orig);
M = QLmaxlev(sizeOrig, filtername);
disp([' Maximum number of scales is ' num2str(M)]);
 Maximum number of scales is 12
N = min(N,M);
disp([' Number of scales is set to ' num2str(N)]);
 Number of scales is set to 12
%
%---DECOMPOSITION--------------------------------------------------------------
[C,S] = QLiftDec2(Orig,N,filtername);
```

retrieveQ1001

Purpose `retrieveQ1001` retrieves detail or approximation coefficients, defined on a quincunx grid (odd slots), from the decomposition according to the lifting scheme.

Syntax `[F10, F01] = retrieveQ1001(level, o, C, S)`

Description `retrieveQ1001` is a utility for the lifting scheme reconstruction. It retrieves detail or approximation coefficients, defined on a quincunx grid (odd slots), from the decomposition vector `C` as created by `QLiftDec2`. The structure and dimensions of decomposition vector `C` and the bookkeeping matrix `S` are supposed to be consistent with how they are produced by `QLiftDec2`.

`level` must be an integer, it indicates the level of `F`. With a decomposition created by `QLiftDec2` an even number for `level` always produces an empty result.

`o` is a character and should be either 'a' or 'd', describing the type of `F`: 'a' relates to approximation (coefficients) and 'd' relates to detail (coefficients).

Both outputs `F10` and `F01` are two-dimensional signals of either detail or approximation coefficients. `F10` corresponds to gridpoints of colour 10, `F01` corresponds to gridpoints of colour 01, see Figure 3. Together, `F10` and `F01` constitute a quincunx grid of odd slots.

See Also `retrieveR`, `retrieveQ0110`, `retrieveQ`.

Example

```
%---PARAMETERS-----------------------------------------------------------------
% How to execute, set parameters
N = 6;                    %  maximum level (even number) in lifting scheme
filtername = 'Neville4';
%
%---INSERT YOUR IMAGE HERE-----------------------------------------------------
load zenithgray; Orig = zenithgray; clear zenithgray;
%
%---DECOMPOSITION--------------------------------------------------------------
[C,S] = QLiftDec2(Orig,N,filtername);
%
%---RETRIEVE DETAIL AT ODD LEVEL-----------------------------------------------
[F10, F01] = retrieveQ1001(N-1, 'd', C, S);
sizeF10 = size(F10)
sizeF01 = size(F01)

sizeF10 =

    74    39


sizeF01 =

    75    38
```

retrieveR

Purpose `retrieveR` retrieves detail or approximation coefficients, defined on a rectangular grid, from the decomposition according to the lifting scheme.

Syntax `F = retrieveR(level, o, C, S)`

Description `retrieveR` is a utility for the lifting scheme reconstruction. It retrieves detail or approximation coefficients, defined on a rectangular grid, from the decomposition vector `C` as created by `QLiftDec2`. The structure and dimensions of decomposition vector `C` and the bookkeeping matrix `S` are supposed to be consistent with how they are produced by `QLiftDec2`.

`level` must be an integer, it indicates the level of `F`. With a decomposition created by `QLiftDec2` an odd number for `level` always produces an empty result.

`o` is a character and should be either 'a' or 'd', describing the type of `F`: 'a' relates to approximation (coefficients) and 'd' relates to detail (coefficients).

Output `F` is a two-dimensional signal of detail or approximation coefficients.

See Also `retrieveQ1001`.

Example

```
%---PARAMETERS-----------------------------------------------------------------
% How to execute, set parameters
N = 6;                    %  maximum level (even number) in lifting scheme
filtername = 'Neville4';
%
%---INSERT YOUR IMAGE HERE-----------------------------------------------------
load zenithgray; Orig = zenithgray; clear zenithgray;
%
%---DECOMPOSITION--------------------------------------------------------------
[C,S] = QLiftDec2(Orig,N,filtername);
%
%---RETRIEVE APPROXIMATION-----------------------------------------------------
A = retrieveR(N, 'a', C, S);
sizeA = size(A)
%
%---RETRIEVE DETAIL AT EVEN LEVEL----------------------------------------------
D = retrieveR(N-2, 'd', C, S);
sizeD = size(D)

sizeA =

    75    39


sizeD =

   148    77
```

See Section 4 for a more involved example.

rota1001fill

Purpose `rota1001fill` rotates and maps the quincunx grid onto a square grid together with the corresponding values. The excess area is filled by padding.

Syntax

```
Q = rota1001fill(F10, F01, bgval)

Q = rota1001fill(F10, F01)
```

Description `rota1001fill` is a utility for mapping two-dimensional signals from a quincunx grid (odd slots) onto a rectangular grid. It facilitates imaging.

Both inputs `F10` and `F01` are two-dimensional signals on a rectangular grid. `F10` corresponds to gridpoints of colour 10, `F01` corresponds to gridpoints of colour 01, see Figure 3. Together, `F10` and `F01` constitute a quincunx grid of odd slots.

`bgval` is an optional argument. The excess area in `Q` is filled by padding with this value. If the argument is not used then the very first value of `F10` is used instead. Note that the latter choice doesn't change neither the minimum nor maximum of the union of the input gridfunctions.

See Also `printshop`, `rota0011fill`.

Example See also Figure 6.

```
%---PARAMETERS-----------------------------------------------------------------
 N=2;
 filtername='maxmin';
%
% Manage output, set preferences: see printshop
 dops   =3;
 dotiff =0;
%---INSERT YOUR IMAGE HERE-----------------------------------------------------
load zenithgray; Orig = zenithgray; clear zenithgray;
%-------------------------------------------------------------------------------
% Some preliminaries
disp([' Filter type is ' filtername]);
disp([' Number of scales asked for is ' num2str(N)]);
%
% Show original image
printshop('figOriginal', ' Original ', Orig, dops, dotiff, []);
%
% Decomposition
[C,S] = QLiftDec2(Orig,N,filtername);
%
% Show the details (at level 1) on the rotated quincunx grid
[Detail10, Detail01] = retrieveQ1001(1, 'd', C, S);
background=max(max(max(Detail10)), max(max(Detail01)));
RotaDet=rota1001fill(Detail10, Detail01, background);
printshop('figDetail', 'Detail', RotaDet, dops, dotiff, []);
%
Approx = retrieveR(N, 'a', C, S);
printshop('figApprox', 'Approximation', Approx, dops, dotiff, []);
```

Figure 6: Original, detail and approximation images.

Q0011mupq

Purpose Computes the central moments with prescribed order of a two-dimensional signal (image, matrix) on a quincunx grid (even slots).

Syntax

```
val = Q0011mupq(F00, F11, p, q)

val = Q0011mupq(F00, F11, p, q, Center)
```

Description `Q0011mupq` is a utility for the computation of moments and invariants. It computes the central moments μ_{pq} of order `p` in the x-direction and of order `q` in the y-direction and the signal `F00` $\cup$ `F11` seen as a density distribution defined on a quincunx grid (even slots). Note that by the values $(p, q) = (1, 0)$ or $(p, q) = (0, 1)$ the central moment has to vanish.

`p` must be an integer, it is the order `p` in the x-direction.

`q` must be an integer, it is the order `q` in the y-direction.

`Center` is an optional argument, it is an array of 2 doubles that is supposed to be the center of mass. Without this argument the center of mass is computed for by this procedure. See also Hu [4].

Output `val` is the value of the central moment as computed.

Algorithm See Section 2.2.

See Also `mupq`, `Q1001masscenter`, `Q1001gridfdims`, `Q1001xcpowp`, `Q1001ycpowp`, `Q1001mupq`.

Example

```
load zenithgray; Orig = zenithgray; clear zenithgray;
%
%---EXTRACT A QUINCUNX GRIDFUNCTION-----
F00 = getcolor00(Orig);
F11 = getcolor11(Orig);
%
%---COMPUTE MASS------------------------
mu00 = Q0011mupq(F00, F11, 0, 0);
%---NICE CHECK: OUTCOME SHOULD VANISH---
mu10 = Q0011mupq(F00, F11, 1, 0);
%---NICE CHECK: OUTCOME SHOULD VANISH---
mu01 = Q0011mupq(F00, F11, 0, 1);
%
[mu00 mu10 mu01]

ans =

  369.4633   -0.0000    0.0000
```

Compare to the result of the example with procedure `mupq`.

whatcoef2QL

Purpose Finds out how a two-dimensional signal (image, matrix) has been stored in the one-dimensional storage vector of coefficients.

Syntax `[first, last] = whatcoef2QL(level, colorF, o, S);`

Description `whatcoef2QL` is a utility for the lifting scheme reconstruction. It finds out from where (`first`) to where (`last`) a two-dimensional signal (image, matrix) has been stored in the one-dimensional storage vector of coefficients. The said two-dimensional signal is determined by its level `level`, type `o` and in case of a quincunx grid also by its colour `colorF`.

`level` must be an integer, it is the level of the two-dimensional signal.

`colorF` must be a string of characters. Its value is needed only for odd `level`, in that case the two-dimensional signal is defined as one of the two signals on rectangular grids that together constitute the quincunx grid. The possible values read (see Figure 3):
`00` corresponds to gridpoints of colour 00,
`11` corresponds to gridpoints of colour 11,
`01` corresponds to gridpoints of colour 01,
`10` corresponds to gridpoints of colour 10,
`'none'` colour is irrelevant.

`o` is a character and should be either 'a' or 'd', describing the type of the two-dimensional signal: 'a' relates to approximation (coefficients) and 'd' relates to detail (coefficients).

`S` must be a two-dimensional array of integers, its structure should be in accordance with the outcome of `QLiftDec2`.

`first` and `last` are the integer outcomes.

See Also `QLiftDec2`, `QLiftRec2`.

Example

```
%---PARAMETERS-------------------------------------------------------------
% How to execute, set parameters
N = 6;                    %  maximum level (even number) in lifting scheme
filtername = 'Neville4';
%
%---INSERT YOUR IMAGE HERE-------------------------------------------------
load zenithgray; Orig = zenithgray; clear zenithgray;
%
%---DECOMPOSITION----------------------------------------------------------
[C,S] = QLiftDec2(Orig,N,filtername);
%
%---FIND OUT HOW DETAIL WITH COLOUR 10 AT ODD LEVEL HAS BEEN STORED---------
[first, last] = whatcoef2QL(N-1, '10', 'd', S);
```

4. Example with image fusion

We present an example of an application of the toolbox to image fusion. Two similar though different images are fused into one image that is meant to unify the information included in both originals. The example follows Li et al. [7], it is not claimed to represent the present state of the art in image fusion. See also Figure 7 for the outcome of the example.

```
%---PARAMETERS-----------------------------------------------------------------
% How to execute, set parameters
%
N=20;
%
%filtername='maxmin';
%filtername='Neville2';
 filtername='Neville4';
%filtername='Neville6';
%filtername='Neville8';
%
% Manage output, set preferences: see printshop
dops   =3;
dotiff =0;
%
%---INSERT YOUR IMAGES HERE-----------------------------------------------------
%
% Convert file with .tif format into matrix of grayvalues
A = double(imread('hoed_A.tif','tiff'));
B = double(imread('hoed_B.tif','tiff'));
%-------------------------------------------------------------------------------
% Some preliminaries
disp([' Filter type is ' filtername]);
disp([' Number of scales asked for is ' num2str(N)]);
%
M = QLmaxlev(size(A), filtername);
disp([' Maximum number of scales is ' num2str(M)]);
N = min(N,M);
disp([' Number of scales is set to ' num2str(N)]);
%
% Show images that have to be fused
txt = [' Original A'];
printshop('figOriginalA', txt, A, dops, dotiff, []);
txt = [' Original B'];
printshop('figOriginalB', txt, B, dops, dotiff, []);
%
% A and B should have identical dimensions
if ~all(size(B) == size(A))
  error(' Dimensions of A and B not consistent ');
end
%
% Decompositions
%
[CA,SA] = QLiftDec2(A,N,filtername);
[CB,SB] = QLiftDec2(B,N,filtername);
```

```
%
% Create C from CA and CB (only the details)
%
C = CA;  S = SA;
for level = 1:N
   rectgrids = mod(level, 2)+1;
   if rectgrids == 2
     [F1A, F2A] = retrieveQ1001(level, 'd', CA, SA);
     [F1B, F2B] = retrieveQ1001(level, 'd', CB, SB);
   else
     F1A = retrieveR(level, 'd', CA, SA);
     F1B = retrieveR(level, 'd', CB, SB);
   end
   for no = 1:rectgrids
      if no == 1
        fca = F1A; fcb = F1B;
      else
        fca = F2A; fcb = F2B;
      end
%
%     A very simple, pixel-based fusion rule
      D = (abs(fca)>abs(fcb));
      fcab = D.*fca + (~D).*fcb;
%
      if rectgrids == 2
        if no == 1
          [first, last] = whatcoef2QL(level, '10', 'd', S);
        else
          [first, last] = whatcoef2QL(level, '01', 'd', S);
        end
      else
        [first, last] = whatcoef2QL(level, 'none', 'd', S);
      end
      C(first:last) = fcab;
   end
end
%
% Update C from CA and CB (only the approximation)
%
fca = retrieveR(level, 'a', CA, SA);
fcb = retrieveR(level, 'a', CB, SB);
fcab = (fca + fcb) * 0.5;
%
[first, last] = whatcoef2QL(level, 'none', 'a', S);
C(first:last) = fcab;
%
% Reconstruction
%
ABfused = QLiftRec2(C,S,filtername);
%
% Show result of fusion
```

```
txt = [' Fusion of A and B '];
printshop('figABfused', txt, ABfused, dops, dotiff, [], 0, 255);
```

Figure 7: Top: input images; below: result of image fusion.

5. Concluding Remarks

The toolbox can be downloaded from the following URL:

`http://www.cwi.nl/ftp/pauldz/Codes/LISQ/`

Any result obtained by this code or part of it, and published in any publication whatsoever, electronically or otherwise, has to make reference to the report you are now reading.

References

1. I. Daubechies and W. Sweldens, Factoring wavelet transforms into lifting steps, *J. Fourier Anal. Appl.*, 4(3), 345–267, 1998.

2. H. Foerster, K. Stüben and U. Trottenberg, Nonstandard multigrid techniques using checkered relaxations and intermediate grids, in *Elliptic Problem Solvers*, M. Schultz (ed.), Academic Press, New York, 1981, 285-300.
3. H.J.A.M. Heijmans and J. Goutsias, Multiresolution signal decomposition schemes. Part 2: Morphological wavelets, CWI Report PNA-R9905, Centrum voor Wiskunde en Informatica, Amsterdam, 1999.
`www.cwi.nl/ftp/CWIreports/PNA/PNA-R9905.pdf`
4. M. Hu, Visual pattern recognition by moment invariants, *IRE Trans. Inf. Th.*, IT-8, 179–187, 1962.
5. A. Jensen and A. la Cour-Harbo, *Ripples in Mathematics: the Discrete Wavelet Transform*, Springer-Verlag Berlin Heidelberg 2001.
6. J. Kovacevic and W. Sweldens, Wavelet families of increasing order in arbitrary dimensions, *IEEE Trans. Imag. Proc.*, 9(3), 480–496, 2000.
7. H. Li and B.S. Manjunath and S.K. Mitra, Multisensor image fusion using the wavelet transform, *Graphical Models and Image Processing* 57 (3): 235–245 (1995).
8. M. Misiti, Y. Misiti, G. Oppenheim, J.-M. Poggi, *Wavelet Toolbox User's Guide (For Use with Matlab)* The MathWorks Inc., 1996.
9. P.J. Oonincx and P.M. de Zeeuw, An Image Retrieval System Based on Adaptive Wavelet Lifting, CWI Report PNA-R0208, Centrum voor Wiskunde en Informatica, Amsterdam, 2002.
`www.cwi.nl/ftp/CWIreports/PNA/PNA-R0208.pdf`
10. W. Sweldens, The lifting scheme: A Custom Design Construction of Biorthogonal Wavelets, *Appl. Comput. Harmon. Anal.* Vol. 3 No.2 1998.
11. W. Sweldens, The lifting scheme: A construction of second generation wavelets, *SIAM J. Math. Anal.*, 29(2), 511–546, 1997.
12. W. Sweldens and Peter Schröder, Building your own wavelets at home, *Wavelets in Computer Graphics*, ACM SIGGRAPH Course notes, 15–87, 1996.
`http://cm.bell-labs.com/who/wim/papers/athome.pdf`
13. D.B.H. Tay and N.G. Kingbury, Flexible design of Multidimensional Perfect Reconstruction FIR two-band filters using Transformation of Variables, *IEEE Trans. Image Processing,* Vol 2, No. 4. October 1993.
14. D.B.H. Tay and N.G. Kingbury, Design of Two-dimensional Perfect Reconstruction Filter Banks using Transformation of Variables : IIR case, *IEEE Trans. Circuits and Systems* Part II. Vol 43, No. 3. March 1996.
15. G. Uytterhoeven and A. Bultheel, The red-black wavelet transform, TW Report 271, Dept. Comp. Sc., Katholieke Universiteit Leuven, Leuven, 1997.
`www.cs.kuleuven.ac.be/publicaties/rapporten/tw/TW271.ps.gz`
16. G. Uytterhoeven and A. Bultheel, The red-black wavelet transform, *Proceedings of IEEE Benelux Signal Processing Symposium,* March 1998 (pp. 191–194).
17. G. Uytterhoeven and A. Bultheel, The red-black wavelet transform and the lifting scheme, TW Report 318, Dept. Comp. Sc., Katholieke Universiteit Leuven, Leuven, 2000.
`www.cs.kuleuven.ac.be/publicaties/rapporten/tw/TW318.ps.gz`