Recursive Filtering in Image Processing

Martin Vicanek

25. March 2016

1 Introduction

Filtering is an important and much used discipline in image processing. The goal is either to remove unwanted components such as noise, or to enhance certain features, or simply as an artistic modification. Some filters act isotropically (gaussian blur), others have a preference with respect to some direction. In this article, a class of simple recursive filters in 2 dimensions are analyzed and examples are given for isotropic as well as directional filters. The latter includes a notch filter for removal of periodic, oriented patterns.

2 Recursive filtering of images

Recursive filters are more efficient than straight convolutions even with moderate kernel sizes. A simple blur with a 100x100 pixels kernel, if implemented in a naive way, would be extremely slow even with todays advanced harware. Recursive filters can handle such a situation dramatically faster [1].

Sometimes filters can be applied in x- and in y-direction in successive, independent passes. These are separable filters, such as the box blur filter or the gaussian blur filter. It is obvious that a directional filter will not be separable in general. A suitable general recursion scheme may scan the image in horizontal lines, calculating each new pixel value from pixels nearby. Let the source image pixel values be denoted by f_{nm} and g_{nm} for the processed

image. Then the iteration process may be described as

$$g_{nm} = \sum_{\ell k} b_{\ell k}\, f_{n-\ell,m-k} - \sum_{\ell k} a_{\ell k}\, g_{n-\ell,m-k} \tag{1}$$

where the sums over ℓk typically encompass the neighborhood. Note that the second term on the right hand side of equation (1) involves pixels already processed. Hence the sum can only include pixels already visited. This is different for the first sum over f_{nm}.

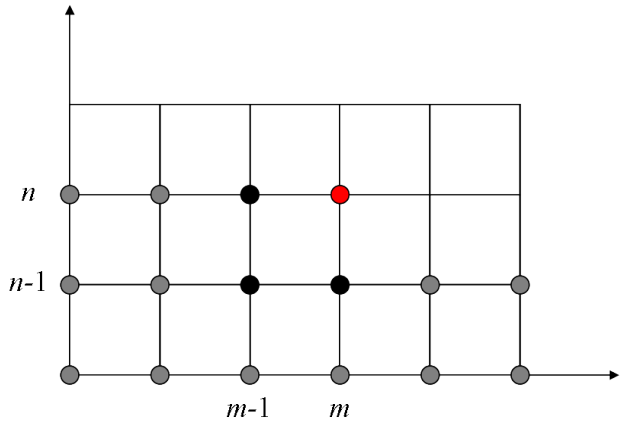

Figure 1: Grid points for image filtering. The red point is current. Gray and black points have been visited. Black points are processed nearest neighbors.

Equation (1) has to be completed by boundary conditions, where not all neigbors exist. A common choice is a reflecting boundary condition, where exterior pixels are generated by mirroring interior pixels at the image boundaries. Another option is to simply replicate the boundary pixels to the exterior. Less suited are periodic boundary conditions, unless the goal is a tileable image.

Filters with constant coefficients are efficiently characterized by their action on individual frequency components of the source. This assumes an expansion of the form

$$f_{nm} = \sum_{\omega_1,\omega_2} F(\omega_1,\omega_2) e^{im\omega_1 + in\omega_2} \tag{2}$$

In this representation, the action of the filter is a simple multiplication,

$$g_{nm} = \sum_{\omega_1,\omega_2} H(\omega_1,\omega_2) F(\omega_1,\omega_2) e^{im\omega_1 + in\omega_2} \tag{3}$$

with the transfer function

$$H(\omega_1, \omega_2) = \frac{\sum_{\ell k} b_{\ell k} e^{-ik\omega_1 - i\ell\omega_2}}{\sum_{\ell k} a_{\ell k} e^{-ik\omega_1 - i\ell\omega_2}} =: \frac{P}{Q}. \tag{4}$$

Recursive filters will introduce a shift in the direction of processing. To compensate for this, filtering is usually applied again in oposite direction, resulting in a zero phase transfer function $|H(\omega_1, \omega_2)|^2$.

Stability analysis of linear recursion equations is well developed in 1D. The matter is more complex for equation (1). For separable 2D filters, the problem may be reduced to 1D.

2.1 The simplest 1D recursive filter

THe simplest recursive filter is a 1D leaky integrator:

$$a_0\, g_{nm} = f_{nm} - a_1\, g_{n,m-1}. \tag{5}$$

Its transfer funcion is $H(\omega_1) = 1/Q$ with

$$Q(\omega_1) = a_0 + a_1 e^{-i\omega_1}. \tag{6}$$

A normalization condition applies to conserve overall brightness

$$a_0 + a_1 = 1. \tag{7}$$

In order to obtain a zero phase filter the iteration is applied twice (forward/backward), resulting in

$$|Q|^2 = A_0 + A_1 \cos\omega_1 \tag{8}$$

with

$$A_0 = 1 - A_1 = 1 - 2a_1(1 - a_1), \quad A_1 = 2a_1(1 - a_1). \tag{9}$$

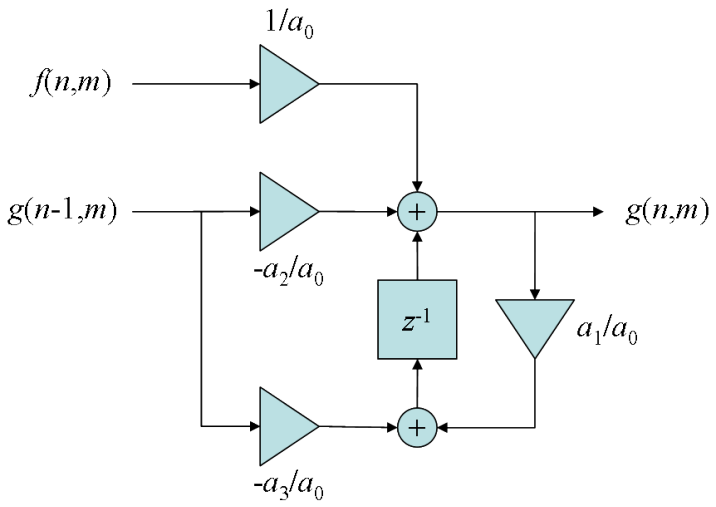

Figure 2: Transposed Direct Form 2 implementation of eq.(14).

2.2 A 4-pass blur filter

It is possible to apply the filter in the previous section first along one axis and then along the other. The overall transfer function is a product of the 1D transfer functions,

$$Q(\omega_1)Q(\omega_2) = (a_0 + a_1 e^{i\omega_1})(a_0 + a_1 e^{i\omega_2}) \tag{10}$$

and

$$|Q_1 Q_2|^2 = A'_0 + A'_1 \cos\omega_1 + A'_2 \cos\omega_2 + A'_3 \cos(\omega_1 + \omega_2) + A'_4 \cos(\omega_2 - \omega_1) \tag{11}$$

with new coefficients A'_n given by

$$A'_0 = A_0^2, \quad A'_1 = A'_2 = 2A_0 A_1, \quad A'_3 = A'_4 = 2A_1^2. \tag{12}$$

To see the filter behavior for long waves, we perform a Taylor expansion near $\omega_1 = \omega_2 = 0$:

$$|Q_1 Q_2|^2 = 1 - \tfrac{1}{2}A_1(\omega_1^2 + \omega_2^2) + \tfrac{1}{24}A_1(\omega_1^4 + \omega_2^4 + 6A_1\omega_1^2\omega_2^2) \tag{13}$$

Note that the filter is isotropic to within 2nd order. It is isotropic to within 4th order for the special case $A_1 = 1/3$. However, A_1 is negative for a blur filter with blur radius $r^2 = -1/A_1$. Below is a pseudocode for an (almost) isotropic 4-pass recursive filter.

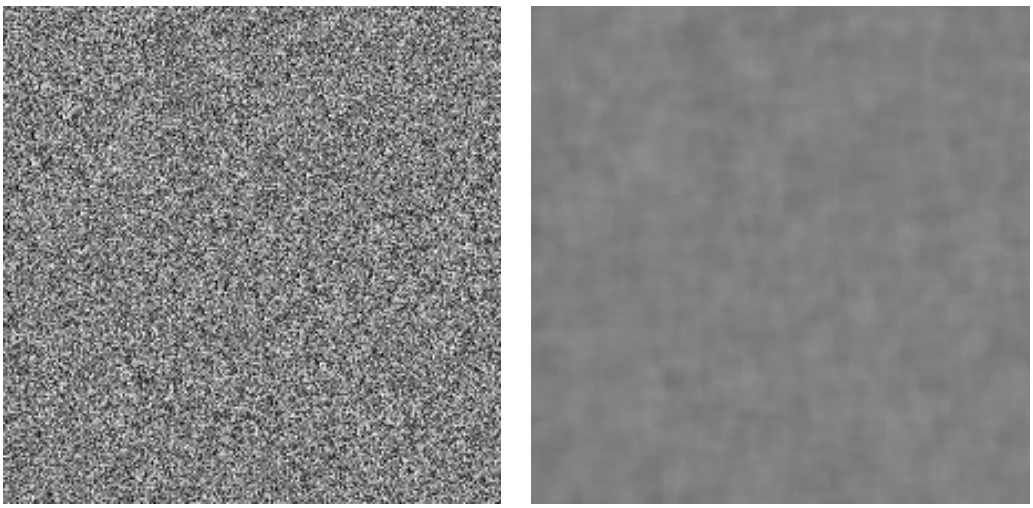

Figure 3: White noise before (left) and after (right) 4-pass filtering.

```
// f(N,M) = N*M image pixel values array
// r = blur radius

a = 2*r/(r + sqrt(r^2 + 2))

// warmup phase reflecting BC
g = 0
do m from M-1 to 0 {
  g = g + a*(f(0,m) - g)
}

do n from 0 to N-1 {
  // 1st pass
  do m from 0 to M-1 {
    g = g + a*(f(n,m) - g)
    f(n,m) = g
  }

  // 2nd pass
  do m from M-1 to 0 {
    g = g + a*(f(n,m) - g)
    f(n,m) = g
  }
}

do m from 0 to M-1 {
```

```
  // 3rd pass
  do n from N-1 to 0 {
    g = g + a*(f(n,m) - g)
    f(n,m) = g
  }

  // 4th pass
  do n from 0 to N-1 {
    g = g + a*(f(n,m) - g)
    f(n,m) = g
  }
}
```

2.3 A 2-pass blur filter

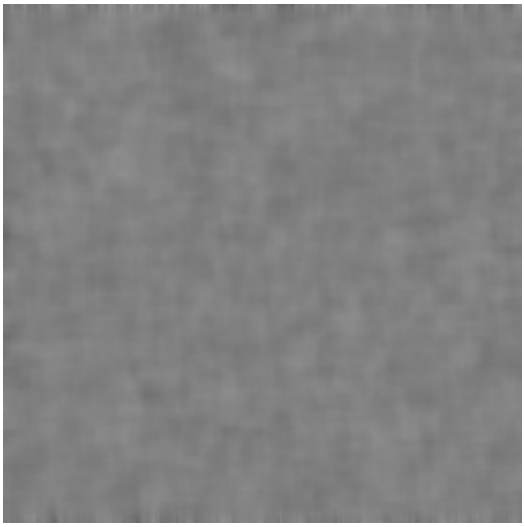

Figure 4: 2-pass blur of white noise. The result is identical to the 4-pass blur except near the image border.

Next consider a 2D filter involving nearest neighbors in the 3rd quadrant, as depicted in figure 1,

$$a_0\, g_{nm} = f_{nm} - a_1\, g_{n,m-1} - a_2\, g_{n-1,m} - a_3\, g_{n-1,m-1} \tag{14}$$

The orresponding transfer funcion is $H = 1/Q$, with

$$Q(\omega_1, \omega_2) = a_0 + a_1 e^{-i\omega_1} + a_2 e^{-i\omega_2} + a_3 e^{-i(\omega_1+\omega_2)}. \tag{15}$$

The normalization codition to conserve overall brightness is

$$a_0 + a_1 + a_2 + a_3 = 1. \tag{16}$$

We obtain a zero-phase filter by forward/backward iteration, resulting in

$$|Q|^2 = A_0 + A_1 \cos\omega_1 + A_2 \cos\omega_2 + A_3 \cos(\omega_1 + \omega_2) + A_4 \cos(\omega_2 - \omega_1). \tag{17}$$

The coefficients A_n are related to the original coefficients a_n of eq. (14) by

$$\begin{aligned} A_0 &= a_0^2 + a_1^2 + a_2^2 + a_3^2 \\ A_1 &= 2(a_0 a_1 + a_2 a_3) & A_2 &= 2(a_0 a_2 + a_1 a_3) \\ A_3 &= 2a_0 a_3 & A_4 &= 2a_1 a_2 \end{aligned} \tag{18}$$

If we want the resulting filter to be isotropic, we need to set

$$A_1 = A_2, \quad A_3 = A_4. \tag{19}$$

Then Taylor expansion near $\omega_1 = \omega_2 = 0$ yields

$$|Q_1 Q_2|^2 = 1 - \tfrac{1}{2}(A_1 + 2A_3)(\omega_1^2 + \omega_2^2) + \tfrac{1}{24}\big[(A_1 + 2A_3)(\omega_1^4 + \omega_2^4) + 12A_3\omega_1^2\omega_2^2\big]. \tag{20}$$

Note that the filter is isotropic to within 2nd order. It is isotropic to within 4th order for the special case $A_1 = 2A_3$.

In terms of the original coefficients, eq.(19) holds if $a_1 = a_2$ and $a_0 a_3 = a_1^2$. This is fulfilled by the following parametric representation,

$$a_0 = 1/(1-q)^2, \quad a_1 = a_2 = -q a_0, \quad a_3 = q^2 a_0. \tag{21}$$

There are other solutions to eq.(19) which, however, lead to unstable recursions. Below is a pseudocode for an (almost) isotropic 2-pass filter.

```
// f(N,M) = N*M image pixel values array
// r = blur radius

q = r^2/(r^2 + 1 + r*sqrt(2*r^2 + 1))
b = (1 - q)^2

// 1st pass
do n from 1 to N-1 {
  g = (1 - q)*f(n,0)
```

```
  do m from 0 to M-1 {
g = b*f(n,m) + q*g
f(n,m) = g + q*f(n-1,m)
  }
}

// 2nd pass
do n from N-2 to 0 {
  g = (1 - q)*f(n,M-1)

  do m from M-1 to 0 {
    g = b*f(n,m) + q*g
    f(n,m) = g + q*f(n+1,m)
  }
}
```

3 Directional blur

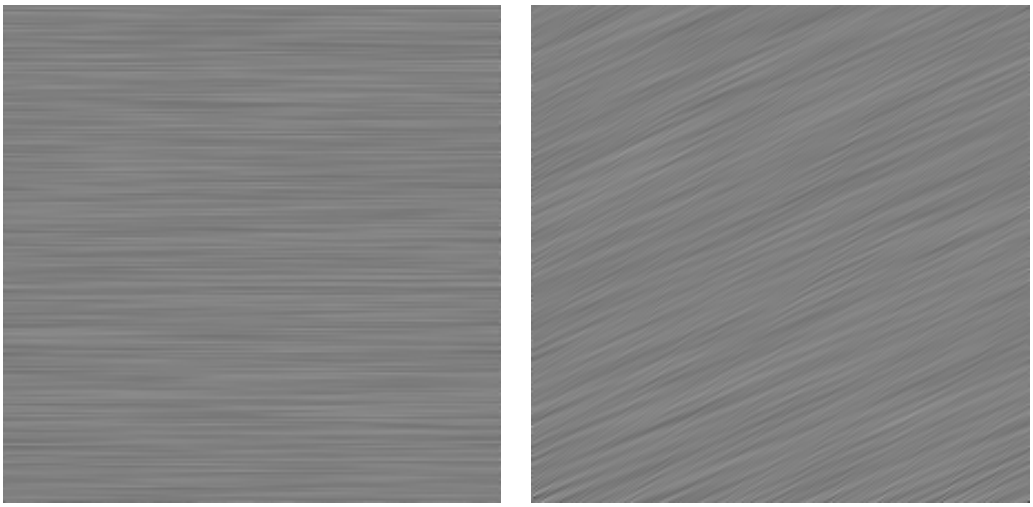

Figure 5: White noise after directional blur. Left: $\theta = 0$. Right: $\theta = 20°$.

In this section we will design a directional blur filter. Ideally, such a filter would act only in one direction and not perpendicular to it. A well-known application is the creation of a brushed aluminum texture from noise, another is simulation of motion blur.

It is fairly easy to design a directional filter if the orientation coincides with one of the image axes. The case of an arbitrary direction is more interesting, however.

Let the direction of blur be characterized by angle θ with respect to the ω_1-axis. Introduce the following abreviations,

$$\mu = \cos\theta, \quad \nu = \sin\theta. \tag{22}$$

A suitable directional lowpass filter function is

$$|Q|^2 = 1 + (\mu\omega_1 + \nu\omega_2)^2 r^2, \tag{23}$$

where r denotes the blur radius.

Taylor expand eq.(17) around $(\omega_1, \omega_2) = (0, 0)$ and equate coefficients up to 2nd order to get

$$\begin{aligned} A_1 + A_3 + A_4 &= -2\mu^2 r^2 \\ A_2 + A_3 + A_4 &= -2\nu^2 r^2 \\ A_3 - A_4 &= -2\mu\nu r^2 \end{aligned} \tag{24}$$

Together with the normalization condition, eq. (16), these are four conditions for four parameters. We may rewrite equations (24) in terms of the original coefficeients a_n using eq.(18),

$$\begin{aligned} a_0a_1 + a_2a_3 + a_0a_3 + a_1a_2 &= -\mu^2 r^2 \\ a_0a_2 + a_1a_3 + a_0a_3 + a_1a_2 &= -\nu^2 r^2 \\ a_0a_3 - a_1a_2 &= -\mu\nu r^2 \end{aligned} \tag{25}$$

This system may be solved in closed form. Write

$$\begin{aligned} (a_0 + a_2)(a_1 + a_3) &= -\mu^2 r^2 = (a_0 + a_2)[1 - (a_0 + a_2)] \\ (a_0 + a_1)(a_2 + a_3) &= -\nu^2 r^2 = (a_0 + a_1)[1 - (a_0 + a_1)] \end{aligned} \tag{26}$$

where the second equality follows from normalization eq.(16). These are quadratic equations for $(a_0 + a_2)$ and $(a_0 + a_1)$, respectively, which may be readily solved,

$$\begin{aligned} a_0 + a_2 &= \tfrac{1}{2} + w_1 \quad w_1 = \sqrt{\tfrac{1}{4} + \mu^2 r^2} \\ a_0 + a_1 &= \tfrac{1}{2} + w_2 \quad w_2 = \sqrt{\tfrac{1}{4} + \nu^2 r^2} \end{aligned} \tag{27}$$

From there, it is easy to solve for the a_n,

$$\begin{aligned} a_1 &= \tfrac{1}{2} + w_2 - a_0 \\ a_2 &= \tfrac{1}{2} + w_1 - a_0 \\ a_3 &= a_0 - w_1 - w_2 \end{aligned} \tag{28}$$

The last equation follows from normalization. Substitute this in the last eq.(25) to get

$$a_0 = (\tfrac{1}{2} + w_1)(\tfrac{1}{2} + w_2) - \mu\nu r^2 \tag{29}$$

Below is a pseudocode for a directional 2-pass filter. Note that the iteration eq.(14), when scanned along a line, may be viewed as a 1d single pole filter with two inputs f_{nm} and $g_{n-1,m}$. The latter is an input because the values have been processed already. We choose the Transposed Direct Form 2 as a suitable implementation, refer to figure 2.

```
// f(N,M) = N*M image pixel values array
// r = blur radius
// angle = direction of blur

mu = cos(angle)
nu = sin(angle)
R1 = (mu*r)^2
R2 = (nu*r)^2
R3 = mu*nu*r^2
w1 = sqrt(0.25 + R1)
w2 = sqrt(0.25 + R2)

a0 = (w1 + 0.5)*(w2 + 0.5) - abs(R3)
a1 = 0.5 + w2 - a0
a2 = 0.5 + w1 - a0
a3 = a0 - w1 - w2

b0 = 1/a0
b1 = -a2/a0
q  = -a1/a0
c  = -a3/a0

// iterate in the direction of blur
if R3 > 0 then
  M0 = 0, M1 = M - 1
else
  M1 = 0, M0 = M - 1
```

```
// 1st pass
do n from 1 to N-1 {

  do m from M0 to M1 {
    f(n,m) = b0*f(n,m) + b1*f(n-1,m) + g
    g = q*f(n,m) + c*f(n-1,m)
  }
}

// 2nd pass
do n from N-2 to 0 {

  do m from M1 to M0 {
    f(n,m) = b0*f(n,m) + b1*f(n+1,m) + g
    g = q*f(n,m) + c*f(n+1,m)
  }
}
```

The 2-pass filter yields a similar blur as the 4-pass filter in the previous section, however there are two differences: (i) it is easier to implement reflecting BC for the 4-pass (ii) the 4-pass is less sensitive to rounding errors. We recommend to use at least 16 bits/channel in both cases.

4 Removal of periodic patterns

Suppose that we want to suppress or entirely remove some of the Fourier components in eq.(2) near some vector frequency $(\omega_1, \omega_2) = (\alpha, \beta)$, e.g. in order to remove a periodic artefact from an image. Since f_{nm} is real valued, $F(\omega_1, \omega_2)$ has Hermitean symmetry. Therefore, we will have to remove components at both (α, β) and $-(\alpha, \beta)$.

We will isolate the unwanted frequencies in three steps: (i) frequency downconversion, (ii) lowpass, and (iii) frequency upconversion. This technique is known in radio signal processing as heterodyning. Here we apply the same technique for image processing in two dimensions.

(i) Frequency downconversion is accomplished by multiplying the image by sine and cosine waves, respectively,

$$f'_{nm} = \cos(n\beta + m\alpha) f_{nm}, \quad f''_{nm} = \sin(n\beta + m\alpha) f_{nm}. \tag{30}$$

Figure 6: Image with periodic artefact, presumably a finger print. Left: before Right: after notch filtering.

(ii) Lowpass filtering an image is essentially a blur. We may use a simple isotropic filter as in section 2.2 or 2.3 to obtain g'_{nm} from f'_{nm} and g''_{nm} from f''_{nm}, respectively,

$$\begin{aligned} g'_{nm} &= \sum_{\omega_1,\omega_2} H(\omega_1,\omega_2) F'(\omega_1,\omega_2) e^{im\omega_1 + in\omega_2} \\ g''_{nm} &= \sum_{\omega_1,\omega_2} H(\omega_1,\omega_2) F''(\omega_1,\omega_2) e^{im\omega_1 + in\omega_2} \end{aligned} \tag{31}$$

The lowpass transfer function H consists of a single peak at $\omega_1 = \omega_2 = 0$.

(iii) Frequency upconversion: consider the expression

$$g_{nm} = 2\cos(n\beta + m\alpha) g'_{nm} + 2\sin(n\beta + m\alpha) g''_{nm}. \tag{32}$$

Then, as shown in the Appendix,

$$g_{nm} = \sum_{\omega_1,\omega_2} \left[H(\omega_1 + \alpha, \omega_2 + \beta) + H(\omega_1 - \alpha, \omega_2 - \beta) \right] F(\omega_1,\omega_2) e^{im\omega_1 + in\omega_2} \tag{33}$$

Hence the result of the three steps (i), (ii) and (iii) is a filter with peaks at frequencies (α, β) and $-(\alpha, \beta)$, respectively. As a consequence, subtracting g_{nm} from the original image f_{nm} results in a directional notch filter.

Below is a pseudocode for a directional notch filter.

```
/*
Input:
  f(N,M) = N*M image pixel values array
  alpha, beta = notch frequencies
  Q = inverse relative notch bandwidth

Output:
  g(N,M) = N*M filtered image pixel values array
*/

// some constants
c1 = cos(alfa)
s1 = sin(alfa)
r  = Q/sqrt(alpha^2 + bea^2)
q  = r^2/(r^2 + 1 + r*sqrt(2*r^2 + 1))
b  = (1 - q)^2

// frequency downconversion
do n from 1 to N-1 {
  c = cos(n*beta)
  s = sin(n*beta)

  do m from 0 to M-1 {
    f1(n,m) = c*f(n,m)
    f2(n,m) = s*f(n,m)
    tmp = c*c1 - s*s1  // generate cos and sin recursively along the way
    s = s*c1 + c*s1
    c = tmp
  }
}

// lowpass (1st pass)
do n from 1 to N-1 {
  g1 = (1 - q)*f1(n,0)
  g2 = (1 - q)*f2(n,0)

  do m from 0 to M-1 {
    g1 = b*f1(n,m) + q*g1
    g2 = b*f2(n,m) + q*g2
    f1(n,m) = g1 + q*f1(n-1,m)
```

```
    f2(n,m) = g2 + q*f2(n-1,m)
  }
}

// lowpass (2nd pass)
do n from N-2 to 0 {
  g1 = (1 - q)*f1(n,M-1)
  g2 = (1 - q)*f2(n,M-1)

  do m from M-1 to 0 {
    g1 = b*f1(n,m) + q*g1
    g2 = b*f2(n,m) + q*g2
    f1(n,m) = g1 + q*f1(n+1,m)
    f2(n,m) = g2 + q*f2(n+1,m)
  }
}

// frequency upconversion
do n from 1 to N-1 {
  c = 2*cos(n*beta)
  s = 2*sin(n*beta)

  do m from 0 to M-1 {
    g(n,m) = f(n,m) - c*f1(n,m) - s*f2(n,m) // subtract for notch
    tmp = c*c1 - s*s1
    s = s*c1 + c*s1
    c = tmp
  }
}
```

5 Conclusion and Outlook

The two examples of directional filters demonstrate that specific modifications can be achieved quite efficiently even with very simple recursive filters. The heterodyning approach to the notch filter seems to be new in image processing.

However, the full potential of recursive filtering may lie beyond linear shift invariance (LSI), i.e. when filter coefficients may vary across the image. Some work has been done for the Gauss filter. In this context it would be interesting to consider other than Direct Forms. From audio processing

we know that certain topologies perform better than others with regard to modulation. Furthermore, other filter forms may have better SNR and low-frequency accuracy.

Nonlinear filters, where filter coefficients are derived from the image itself, include bilateral filtering, anisotropic filtering, edge-aware smoothing, detail enhancement, HDRI.

6 Apendix

We will prove eq.(33). Consider a single spectral component in one dimension only,

$$f_n = e^{in\omega}. \tag{34}$$

Then

$$\begin{aligned} f'_n &= \cos(n\alpha) f_n = \tfrac{1}{2}\big(e^{in(\omega+\alpha)} + e^{in(\omega-\alpha)}\big) \\ f'_n &= \sin(n\alpha) f_n = \tfrac{1}{2i}\big(e^{in(\omega+\alpha)} - e^{in(\omega-\alpha)}\big). \end{aligned} \tag{35}$$

Applying a lowpass filter to these expressions will give mostly zero except in two cases:

Case 1: $\omega \approx \alpha$

$$g'_n = \tfrac{1}{2} H(\omega-\alpha) e^{in(\omega-\alpha)}, \quad g''_n = -\tfrac{1}{2i} H(\omega-\alpha) e^{in(\omega-\alpha)}. \tag{36}$$

Now multiply again with sine and cosine waves,

$$\begin{aligned} 2\cos(n\alpha) g'_n &= \tfrac{1}{2} H(\omega-\alpha) e^{in(\omega-\alpha)}\big(e^{in\alpha} + e^{-in\alpha}\big) \\ 2\sin(n\alpha) g''_n &= -\tfrac{1}{2} H(\omega-\alpha) e^{in(\omega-\alpha)}\big(e^{in\alpha} - e^{-in\alpha}\big). \end{aligned} \tag{37}$$

Adding the two expressions, two terms cancel and the result is

$$g_n = H(\omega-\alpha) e^{in\omega}, \quad \omega \approx \alpha. \tag{38}$$

Case 2: $\omega \approx -\alpha$

$$g'_n = \tfrac{1}{2} H(\omega+\alpha) e^{in(\omega+\alpha)}, \quad g''_n = \tfrac{1}{2i} H(\omega+\alpha) e^{in(\omega+\alpha)} \tag{39}$$

Multiply again with sine and cosine waves,

$$\begin{aligned} 2\cos(n\alpha)g'_n &= \tfrac{1}{2}H(\omega+\alpha)e^{in(\omega+\alpha)}\big(e^{in\alpha}+e^{-in\alpha}\big) \\ 2\sin(n\alpha)g''_n &= -\tfrac{1}{2}H(\omega+\alpha)e^{in(\omega+\alpha)}\big(e^{in\alpha}-e^{-in\alpha}\big). \end{aligned} \tag{40}$$

Adding the two expressions, again two terms cancel and the result is

$$g_n = H(\omega+\alpha)e^{in\omega}, \quad \omega \approx -\alpha. \tag{41}$$

If the lowpass filter is sufficiently narrow, the two above cases are mutually exclusive and we may write

$$g_n = \big[H(\omega-\alpha)+H(\omega+\alpha)\big]e^{in\omega}. \tag{42}$$

Equation (42) holds for a single Fourier component. However, since all operations are linear, the result may be generalized for any linear combination $f_n = \sum_\omega F(\omega)e^{in\omega}$,

$$g_n = \sum_\omega \big[H(\omega-\alpha)+H(\omega+\alpha)\big]F(\omega)e^{in\omega} \tag{43}$$

Eqquation (43) is the result for one dimension. Generalization to two dimentions is straght forward, as the operations on the different dimensions are largely independent. (For a separable lowpass filter kernel, decoupling is complete.) The result is given in equation (33).

References

[1] van Vliet LJ, Young IT, Verbeek PW. Recursive Gaussian derivative filters. In: Proceedings of the 14th International Conference on Pattern Recognition, Brisbane, Australia, August 1998. p. 50914.