accessing pixel values outside the support of the image. Therefore, a method must be used to impute pixel values in the vicinity of the image boundaries, *i.e.*, to generate an extended image $f_{\text{ext}}[\boldsymbol{k}]$ that includes a margin with a width/height/depth of $\lfloor M/2-1 \rfloor$. Four common padding methods are described in the following subsections and are compared qualitatively in Fig. 2.2 and 2.3, where a CT image of honeycombing lung parenchyma (Fig. 2.2a, $N = 301$) is smoothed by convolution with an isotropic Gaussian filter (Fig. 2.2b, $M = 70$, $\sigma = 15$).

Because all padding methods make arbitrary assumptions concerning image content beyond the boundaries, a method should be chosen based on the expected image background. However, general advantages and disadvantages exist and are discussed in the next sections.

2.2.1 Constant Value Padding Z3VE

The simplest method to extend the support of $f[\boldsymbol{k}]$ is to pad it with a constant value C as

$$\begin{cases} f_{\text{ext}}^{\text{constant}}[\boldsymbol{k}] = f[\boldsymbol{k}] & \forall \quad (k_1, \ldots, k_D) \in \{0, \ldots, N-1\}^D \quad \text{and,} \\ f_{\text{ext}}^{\text{constant}}[\boldsymbol{k}] = C & \text{otherwise.} \end{cases}$$

Thus, all pixels outside the original image are assigned the constant value C. This is illustrated in Fig. 2.2c. Though constant value padding is simple to implement, potentially sharp transitions between C and the image value at a boundary may yield inconsistent filter responses. Distortions may appear in the boundary region of the response image as a consequence. Such artefacts can be observed in all boundaries of the filtered image in Fig. 2.2d. With $C = 0$, this method is also called *zero padding*.

2.2.2 Nearest Value Padding SIJG

The nearest value padding method consists of repeating the values of the image at the boundary. We have

$$\begin{cases} f_{\text{ext}}^{\text{nearest}}[\boldsymbol{k}] = f[\boldsymbol{k}] & \forall \quad (k_1, \ldots, k_D) \in \{0, \ldots, N-1\}^D \quad \text{and,} \\ f_{\text{ext}}^{\text{nearest}}[\boldsymbol{k}] = f[\boldsymbol{k}_{\text{nearest}}] & \text{otherwise,} \end{cases}$$

where $\boldsymbol{k}_{\text{nearest}} := \underset{\boldsymbol{k}_{\text{nearest}}}{\arg\min}\left(||\boldsymbol{k} - \boldsymbol{k}_{\text{nearest}}|| \right)$ is the closest neighbour in f. Thus, all pixels outside the original image get the intensity value of the closest pixel in the original image. This is illustrated in Fig. 2.2e. The advantage of this method is that the transitions at the boundary are usually relatively small. However, it introduces a nonexistent pattern in the response image. This method is also called *replicate*.

2.2.3 Periodisation Z7YO

Another straightforward method is to repeat the image along every dimension, *i.e.*, so to periodise the image content. The extended image is therefore equivalent to the original image modulo its support as

$$f_{\text{ext}}^{\text{periodise}}[\boldsymbol{k}] = f[k_1 \mod N, \ \ldots, \ k_D \mod N]$$

for indices $k_i = 0, \ldots, N-1$. This is illustrated in Fig. 2.3a. The introduced patterns are consistent with the actual image content. However, strong transitions are possible at the boundaries. The subsequent artefacts can be observed in the upper part of the left boundary of the example response image (see Fig. 2.3b). It is worth noting that this periodisation is implicit if the convolution operation is performed in the Fourier domain using Eq. (2.3). This method is also called *wrapping*,

circular or *tiling*.

2.2.4 Mirror zDTV

v5: Clarified use of mirror boundary condition by the IBSI.

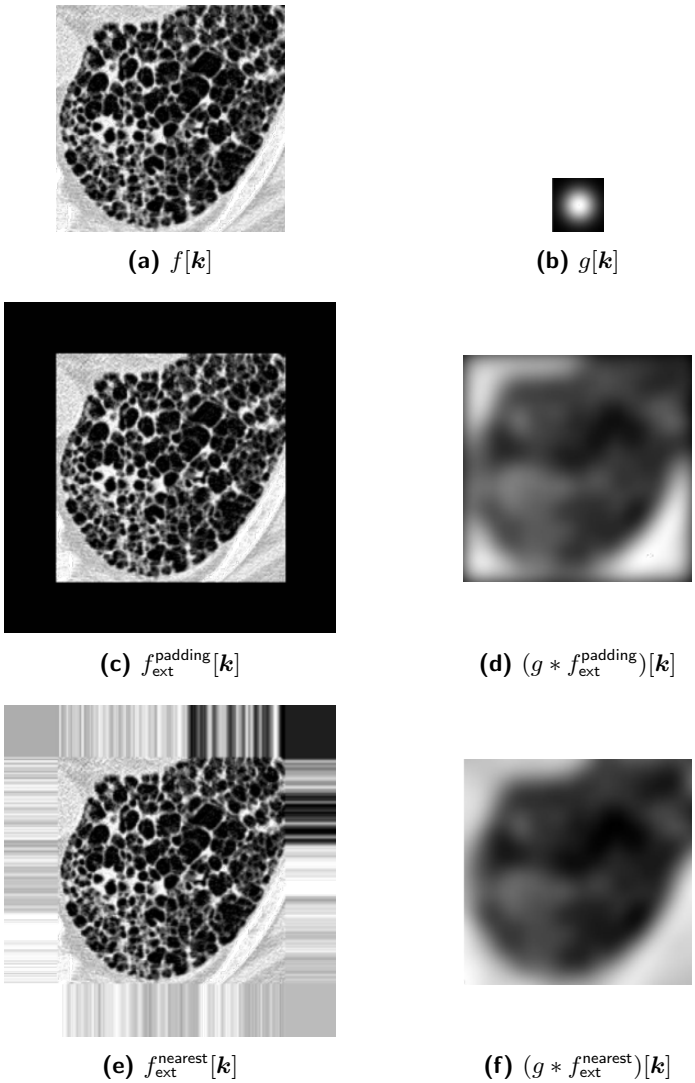
The mirror folding method consists of symmetrising the image at the boundaries. The extended image is $f_{\text{ext}}^{\text{mirror}}[\boldsymbol{k}] = f[\boldsymbol{k'}]$, with $\boldsymbol{k'} = (k'_1, \cdots, k'_D) \in \mathbb{Z}^D$, where

$$k'_i = \begin{cases} k_i \mod N & \text{if } \lfloor \frac{k_i}{N} \rfloor \text{ is even,} \\ N - (k_i \mod N + 1) & \text{otherwise,} \end{cases}$$

for indices $k_i = 0, \ldots N-1$.

This is illustrated in Fig. 2.3c. The introduced patterns are consistent with the actual image content and the transitions at the boundaries are relatively smooth, minimising convolution artefacts (see Fig. 2.3d). Depending on the software implementation, the extended image may include or exclude the boundary pixels of the original image. This method is also called *symmetric*.

For consistency reasons, we assume that boundary pixels are included.

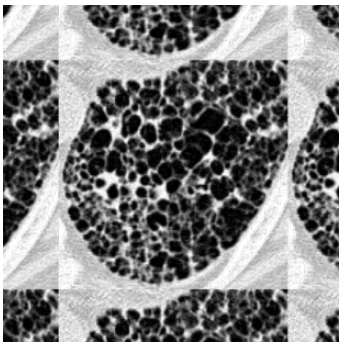


(a) $f[\boldsymbol{k}]$ **(b)** $g[\boldsymbol{k}]$

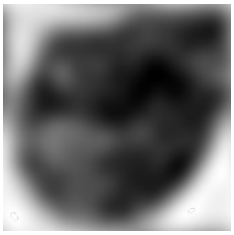
(c) $f_{\mathsf{ext}}^{\mathsf{padding}}[\boldsymbol{k}]$ **(d)** $(g * f_{\mathsf{ext}}^{\mathsf{padding}})[\boldsymbol{k}]$

(e) $f_{\mathsf{ext}}^{\mathsf{nearest}}[\boldsymbol{k}]$ **(f)** $(g * f_{\mathsf{ext}}^{\mathsf{nearest}})[\boldsymbol{k}]$

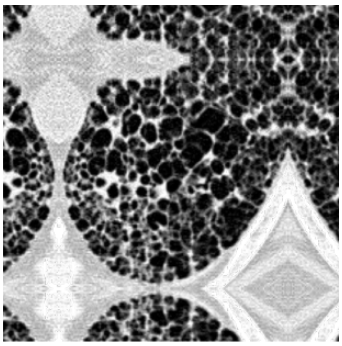
Figure 2.2 — Qualitative comparison of various methods for imputing image values at the boundaries. The image $f[\boldsymbol{k}]$ is smoothed by convolution with the Gaussian filter $g[\boldsymbol{k}]$. The response maps using either zero padding or nearest methods are compared.



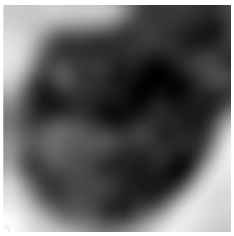
(a) $f_{\text{ext}}^{\text{periodise}}[\boldsymbol{k}]$



(b) $(g * f_{\text{ext}}^{\text{periodise}})[\boldsymbol{k}]$



(c) $f_{\text{ext}}^{\text{mirror}}[\boldsymbol{k}]$



(d) $(g * f_{\text{ext}}^{\text{mirror}})[\boldsymbol{k}]$

Figure 2.3 — Qualitative comparison of various methods for imputing image values at the boundaries. The image $f[\boldsymbol{k}]$ is smoothed by convolution with the Gaussian filter $g[\boldsymbol{k}]$ (see Fig. 2.2). The response maps using either periodisation or mirroring methods are compared.

Chapter 4

Common Convolutional Filters

This section details the definitions of common convolutional filters used for radiomics. Guidance is also provided on how to set their parameters and to obtain appropriate features via matching aggregation functions.

4.1 Identification of Common Approaches

4.2 Mean Filter **S60F**

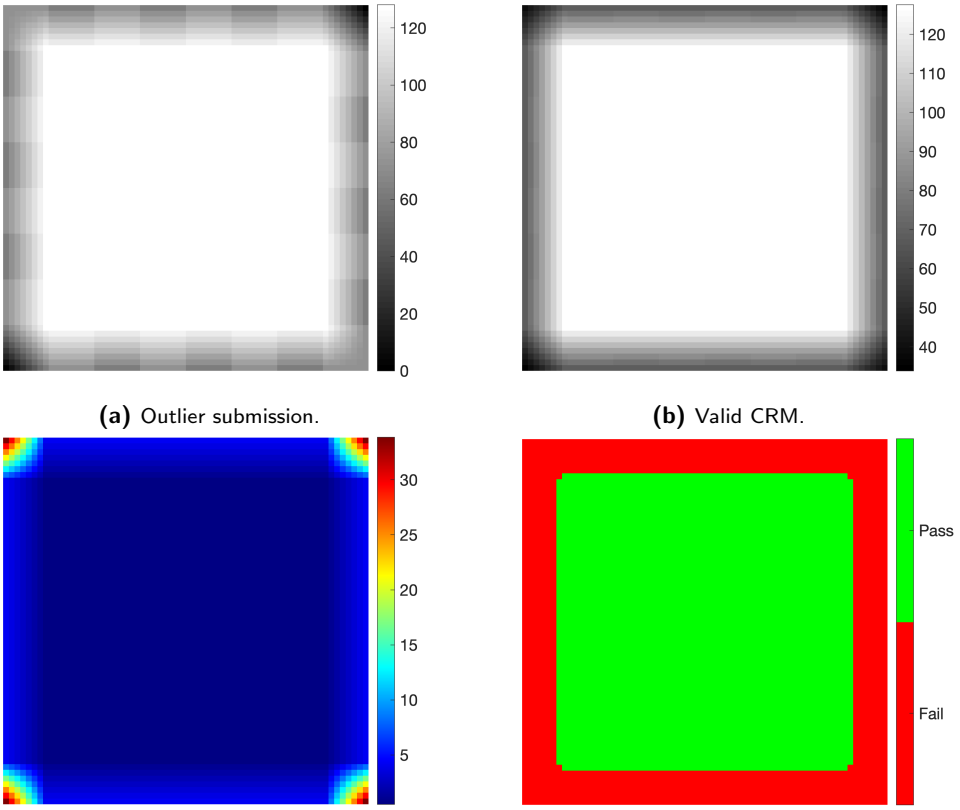
One of the simplest existing kernels is the mean filter that computes the average intensity over its $M \times \cdots \times M$ spatial support[b] (YNOF) as

$$g[\boldsymbol{k}] = \begin{cases} \frac{1}{M^D} & \text{if} \quad k_1, \ldots, k_D \in \left[-\left\lfloor \frac{M}{2} \right\rfloor, \left\lfloor \frac{M}{2} \right\rfloor \right], \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

Implementation Troubleshooting

- Ensure that filter support M is defined in voxel units.
- Check that the padding method is correctly applied (particularly mirror). An example of an outlier submission due to padding is shown in Fig. 4.1.

[a] `http://texrad.com`, as of September 2019.
[b] We restrict the definition to odd values of M.



(a) Outlier submission. **(b)** Valid CRM.

(c) Absolute difference. **(d)** Voxel-wise *passing map*.

Figure 4.1 — Example of outlier discrepancy for mean filter test 1.a.1 (Table 6.1) caused by padding differences. The central 2D slice of the 3D volumes are visualised for: **(a)** an outlier submission, **(b)** the valid *consensus response map* (CRM) found, **(c)** the absolute difference between the outlier and consensus, **(d)** and voxel-wise passing map of this difference.

4.3 Laplacian of Gaussian L6PA

The Laplacian of Gaussian (LoG) is a band-pass and circularly/spherically symmetric convolutional operator. It is therefore invariant to local rotations but also directionally insensitive. Its profile $g[\boldsymbol{k}]$ only depends on the $1D$ radius $||\boldsymbol{k}||$ and corresponds to a radial second-order derivative of a D-dimensional Gaussian filter as

$$g_\sigma[\boldsymbol{k}] = -\frac{1}{\sigma^2}\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^D \left(D - \frac{||\boldsymbol{k}||^2}{\sigma^2}\right) e^{-\frac{||\boldsymbol{k}||^2}{2\sigma^2}}, \tag{4.2}$$

where the standard deviation of the Gaussian σ controls the scale of the operator (41LN). Note that σ is implied to be in voxel units, i.e. $\sigma = \sigma^*/s$, with s the voxel spacing.

LoG filtering cannot be implemented with separable convolution and requires a full D-dimensional convolution. However, it can be approximated using a Difference of two Gaussians (DoG) when the ratio between their respective standard deviations σ_1 and σ_2 is $\sigma_1 = \frac{\sigma_2}{\sqrt{2}}$. Because Gaussian kernels are separable, DoG filtering can also be efficiently implemented using separable convolutions. With an adequate sequence of σ_i, a collection of LoGs can cover the entire image spectrum. In this case, they form wavelets and are often called Mexican hat, Ricker, or Marr wavelet.

The spatial support of the LoG is $(-\infty, \infty)$ (*i.e.* not compact). Because this would require a filter with infinite spatial support, the LoG filter is cropped in practice, usually based on the σ

Figure 4.2 — Examples of $2D$ LoG filters with $\sigma = 16$, $M = 129$ (left) and $\sigma = 8$, $M = 65$ (right).

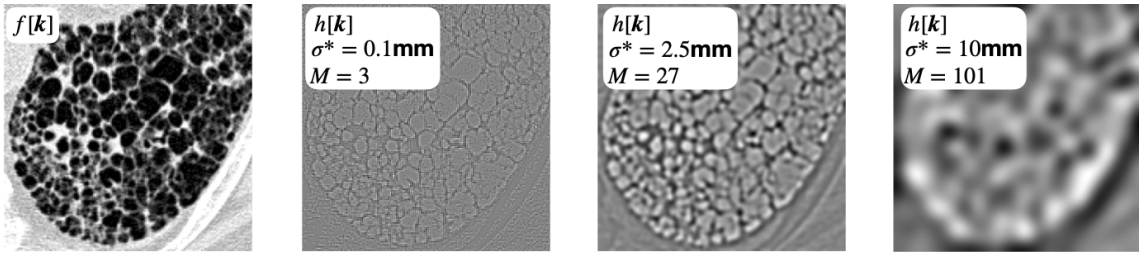


Figure 4.3 — Examples of image filtering with a LoG filter (the pixel spacing is 0.8mm, mirror boundary conditions used for the convolution) at various scales. The small scale ($\sigma^* = 0.1$mm) highlights tiny collagen fibers, whereas the larger scale ($\sigma^* = 10$mm) highlights larger image blobs or clusters present in the diseased lung tissue (honeycombing).

parameter (WGPM). The $1D$ filter size is then

$$M = 1 + 2\lfloor d\,\sigma + 0.5 \rfloor,$$

with d the truncation parameter, σ in voxel units (*e.g.*, $\sigma = 2.5$ voxels when it parameterised as $\sigma^* = 5$ mm with voxel spacing of 2 mm). As a consequence, the size of a ($1D$) LoG filter is at least $M = 1$ and will have an odd, integer value. $d = 4$ is commonly used for truncation.

The profile of a $2D$ LoG is illustrated in Fig. 4.2. The LoG filter can be used to enhance image blobs and ridges at a specific scale, controlled by σ^*. This is illustrated in Fig. 4.3.

Implementation Troubleshooting

- The scale parameter σ^* for the filter tests (Table 6.1) is defined in physical distance. Voxel dimensions for the digital phantoms are stored in millimeters in NifTI or DICOM headers. Many standard implementations assume that σ is defined in voxel units, so a conversion between physical and voxel units is required (e.g. *fspecial3* in Matlab).
- To test implementation of Eq. 4.2, check that the LoG kernel sums to approximately 0.

4.4 Laws Kernels JTXT

Laws kernels are a collection of five types of small $1D$ filters $g[k]$[33]. They are combined using outer products to obtain $2D$ and $3D$ filters (JVAD).

The first type is a low-pass kernel called *Level* for grey level averaging, which is available in two scales with a spatial support of 3 (B5BZ) or 5 (6HRH) pixels:

$$g_{L3}[k] = \frac{1}{\sqrt{6}} \cdot [1, 2, 1]\,, \quad g_{L5}[k] = \frac{1}{\sqrt{70}} \cdot [1, 4, 6, 4, 1]\,.$$

The next kernels are all zero mean, which makes them insensitive to the average grey level. They

$g_{L5S5}[\boldsymbol{k}]$ $g_{S5L5}[\boldsymbol{k}]$ $g_{R5R5}[\boldsymbol{k}]$

Figure 4.4 — Example of $2D$ Laws kernels.

will solely focus on spatial transitions between the values (*i.e.* texture). The four types of transitions are:

- edges:
  - $g_{E3}[k] = \frac{1}{\sqrt{2}} \cdot [-1, 0, 1]$ LJ4T
  - $g_{E5}[k] = \frac{1}{\sqrt{10}} \cdot [-1, -2, 0, 2, 1]$ 2WPV
- spots:
  - $g_{S3}[k] = \frac{1}{\sqrt{6}} \cdot [-1, 2, -1]$ MK5Z
  - $g_{S5}[k] = \frac{1}{\sqrt{6}} \cdot [-1, 0, 2, 0, -1]$ RXA1
- wave:
  - $g_{W5}[k] = \frac{1}{\sqrt{10}} \cdot [-1, 2, 0, -2, 1]$ 4ENO
- ripple:
  - $g_{R5}[k] = \frac{1}{\sqrt{70}} \cdot [1, -4, 6, -4, 1]$ 3A1W

Laws $2D$ and $3D$ kernels are separable by design, and response maps are created by $1D$ kernel convolution along each image direction. Such response maps are referred to by their 1-D kernel names. For instance, a $2D$ response map called h_{L5S5} is obtained after first convolving the image with the g_{L5} kernel along the image lines (*i.e.* k_1) and a subsequent convolution of the image with the g_{S5} kernel along the image columns (*i.e.* k_2). Examples of $2D$ Laws kernels are shown in Fig. 4.4. Note that the above definitions for Laws kernels were normalised, and in this sense deviate from those originally defined by Laws himself[33].

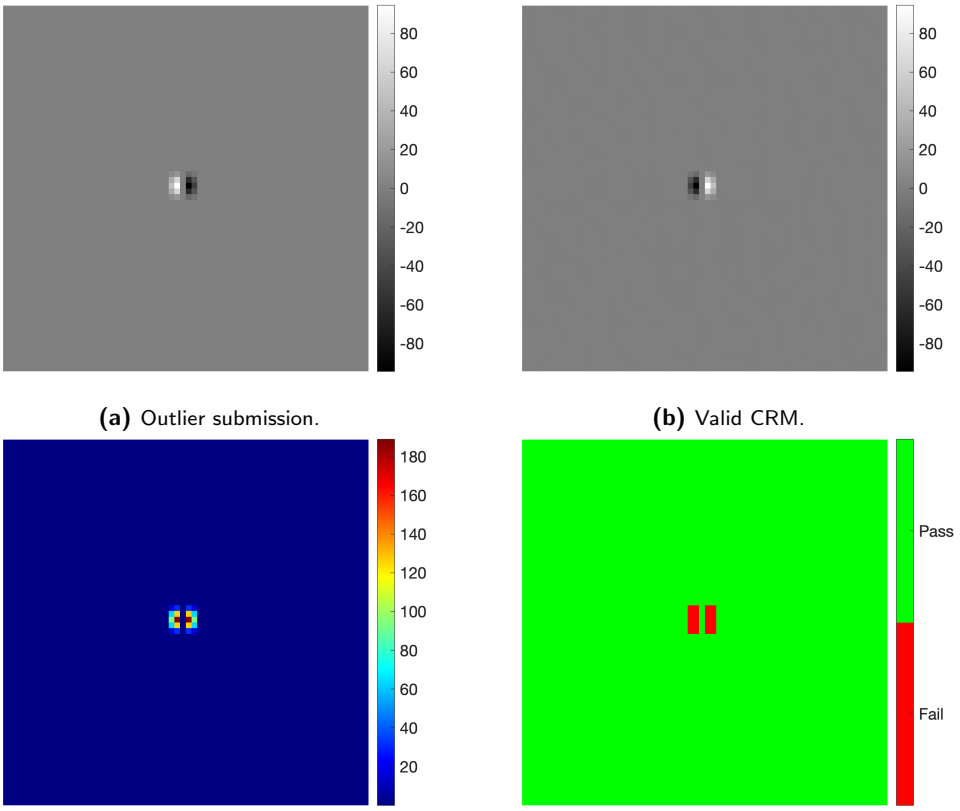
4.4.1 Laws Texture Energy Images PQSD

Laws used his kernels to generate texture energy images[33]. This is done in two steps. In the first step, a response map h is generated by convolving the image with a Laws kernel along each image direction, as described above. Then, a smoothed image is computed where the absolute intensities[c] of voxels in h within Chebyshev distance δ (I176) of a centre voxel are summed to create an energy image h_{energy}:

$$h_{\text{energy}}[\boldsymbol{k}] = \frac{1}{W} \sum_{k_{0,1}=-\delta}^{\delta} \cdots \sum_{k_{0,d}=-\delta}^{\delta} |h[\boldsymbol{k} + \boldsymbol{k}_0]| \,,$$

where $\boldsymbol{k}_0 = (k_{0,1}, \ldots, k_{0,d})$ and $W = (2\delta + 1)^N$ the number of voxels in the N-dimensional neighbourhood. In practice, h_{energy} can be computed using kernel convolutions by convolving $|h|$ with a

[c] It is worth noting that whereas "energy" often involves the computation of squared quantities, the absolute value was proposed by Laws. The goal is to regroup negative and positive filter responses.



(a) Outlier submission. **(b)** Valid CRM.

(c) Absolute difference. **(d)** Voxel-wise *passing map*.

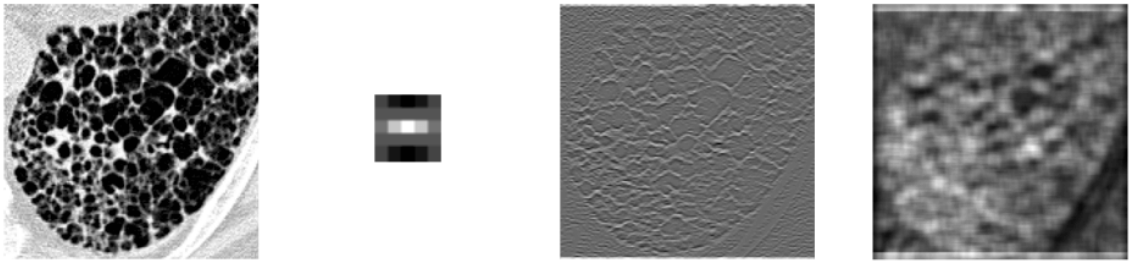
Figure 4.5 — Example of outlier discrepancy for Laws filter test 3.a.1. (Table 6.1) The central 2D slice of the 3D volumes are visualised for: **(a)** an outlier submission, **(b)** the valid *consensus response map* (CRM) found, **(c)** the absolute difference between the outlier and consensus, **(d)** and voxel-wise passing map of this difference. The two response maps (**(a)** and **(b)**) are identical apart from orientation.

$2\delta + 1$ element long $1D$ kernel with constant values $1/(2\delta + 1)$ along each of the image directions, *i.e.* a mean filter as described in Section 4.2.

Note that the definition given above deviates from the one given by Laws[33] by introducing the normalisation factor W. Laws moreover suggested using a sliding window of 15×15 pixels, which corresponds to $\delta = 7$. We recommend that δ is chosen within the context of a given application.

To summarize, Laws filtering requires the following sequence of operations: (i) pad the input image, (ii) filter with a given Laws kernel, (iii) pad the response map and (iv) compute the energy image via the mean filter. We suggest using the same padding (see Section 2.2.1) used to compute the initial response of the Laws kernel to compute the energy image. Laws filtering is not rotation-invariant. However, since the kernels are separable, rotational invariance can be efficiently approximated using permutations and unidimensional filter flipping, followed by orientation pooling with *e.g.* the max (see Section 3.3 and Appendix A). In this case, we recommend computing the texture energy image after orientation pooling.

An example of image filtering with the g_{L5S5} kernel is shown in Fig. 4.6.



$f[\boldsymbol{k}]$ $g_{L5S5}[\boldsymbol{k}]$ $h_{L5S5}[\boldsymbol{k}]$ $h_{\text{energy}}[\boldsymbol{k}]$

Figure 4.6 — Example of image filtering with the g_{L5S5} kernel. The zero padding boundary condition was used to calculate both h_{L5S5} and h_{energy}.

Implementation Troubleshooting

- Filter orientation and convolution direction is a key cause of discrepancy, as some Laws kernels are not symmetric. An example of an outlier response map for filter test 3.a.1 (Table 6.1) that appears to have a different orientation is shown in Fig. 4.5. Check that filter kernels are applied in the direction consistent with Section 1.2.1. Use the orientation phantom to check if your software orients the image as aspected.
- Only one Energy Image should be calculated. This is performed after orientation pooling and on absolute intensity values.

4.5 Gabor Q88H

Gabor filter banks allow for extracting multi-directional and multi-scale texture information via a systematic parcellation of the Fourier domain with elliptic Gaussian windows[6] (see Fig. 3.2 of Depeursinge *et al.*[14]). In the spatial domain, $2D$ Gabor kernels are complex Gaussian-windowed oscillatory functions defined as

$$g_{\sigma,\lambda,\gamma,\theta}[\boldsymbol{k}] = \mathrm{e}^{-\frac{\tilde{k}_1^2+\gamma^2\tilde{k}_2^2}{2\sigma^2}+\mathrm{j}\frac{2\pi\tilde{k}_1}{\lambda}}, \tag{4.3}$$

where σ controls the scale of the filter (standard deviation of the Gaussian envelope; 41LN) and λ is the wavelength (*i.e.* inverse of the frequency $F = \frac{1}{\lambda}$ of the oscillations; S4N6). Both σ and λ are implied to be in voxel units, i.e. $\sigma = \sigma^*/s$ and $\lambda = \lambda^*/s$, with s the voxel spacing, and σ^* and λ^* defined in mm. γ is the spatial aspect ratio (*i.e.* the ellipticity of the support of the filter as $\gamma = a_1/a_2$; GDR5), $(\tilde{k}_1, \tilde{k}_2) = \mathrm{R}_\theta \boldsymbol{k}$ defines the radial and orthoradial elliptic Gaussian axes at the orientation θ (FQER) via the $2D$ rotation matrix $\mathrm{R}_\theta = \begin{pmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{pmatrix}$. As a convention, we define θ to turn clockwise in the axial plane (k_1, k_2). The considered coordinate system follows the conventions introduced in Section 1.2.1. It is depicted in Fig. 4.7 along with an example of a $2D$ Gabor filter seen in the spatial domain.

v4: Clarified units of σ and λ parameters.

In practice, the Gabor function is used as a filter kernel and the spatial frequency bandwidth of the filter needs to be defined. Petkov *et al.*[39] established the relation between the half-response spatial frequency bandwidth F_b (in units of octaves) and the ratio σ/λ as

$$F_b = \log_2 \left(\frac{\frac{\sigma}{\lambda}\pi + \sqrt{\frac{\ln 2}{2}}}{\frac{\sigma}{\lambda}\pi - \sqrt{\frac{\ln 2}{2}}} \right), \text{ and inversely } \quad \frac{\sigma}{\lambda} = \frac{1}{\pi}\sqrt{\frac{\ln 2}{2}} \cdot \frac{2^{F_b}+1}{2^{F_b}-1}. \tag{4.4}$$

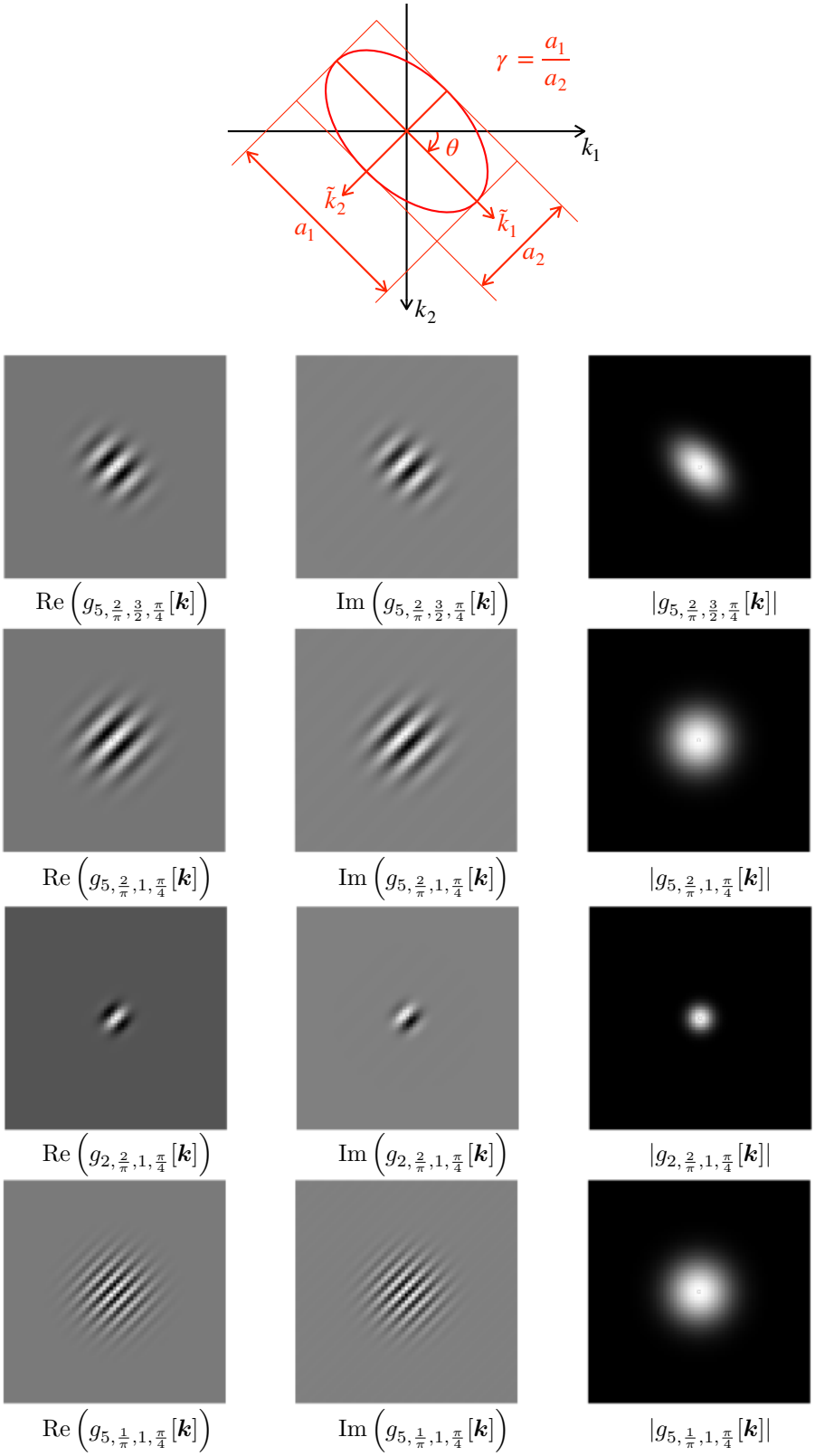


Figure 4.7 — Coordinate system and examples of $2D$ Gabor filters in the spatial domain computed on 65×65 grids with a pixel spacing of 0.8mm.

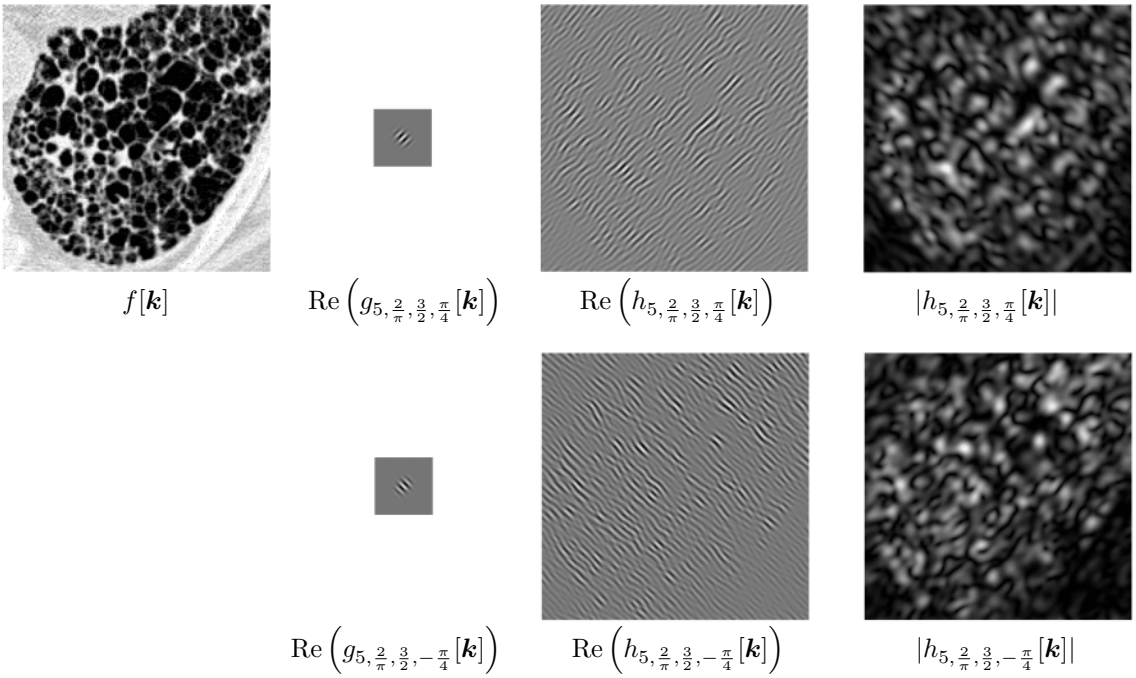


Figure 4.8 — Example of image filtering with Gabor filters (the pixel spacing is 0.8mm, mirror boundary conditions used for the convolution). Collagen junctions oriented at $\theta = -\frac{\pi}{4}$ (top) and $\theta = \frac{\pi}{4}$ (bottom) are highlighted.

Gabor filters can then be constructed at multiple scales and orientations to explore the spectrum of patterns in an image. Bianconi *et al.*[6] proposed to extract response maps at multiple orientations $\{\theta_1, \ldots, \theta_P\}$ and frequencies $\{F_1, \ldots, F_Q\}$ along with γ are defined to cover all directions and scales up to the maximum frequency $F_Q = \frac{1}{\lambda_Q}$. It is worth noting that when using a suitable sequence of scales, Gabor filters can also satisfy the wavelet admissibility condition (*i.e.* referred to as "Gabor wavelets") and can, in this particular case, fully cover the Fourier domain. Finally, since $g_{\sigma,\lambda,\gamma,\theta}[\boldsymbol{k}]$ is complex, the modulus of the associated response map can be used before aggregation and feature calculation as $|h[\boldsymbol{k}]| = |(g_{\sigma,\lambda,\gamma,\theta} * f)[\boldsymbol{k}]|$. An example of filtering with Gabor is depicted in Fig. 4.8.

As with Laplacian-of-Gaussian filters the spatial support of Gabor filters is $(-\infty, \infty)$ (*i.e.* not compact). However, unlike Laplacian-of-Gaussian filters, many standard implementations for Gabor filters do not use truncation. Instead, the most consistent results seem to be achieved by cropping the filter along each dimension i as follows:

v4: Clarified spatial extent of Gabor filter support; v6: Updated description to produce the most consistent results.

$$M_i = \begin{cases} N_i + 1 & \text{if } N_i \text{ is even} \\ N_i & \text{if } N_i \text{ is odd} \end{cases} \tag{4.5}$$

Gabor filters are not rotation-invariant and are therefore best suited for applications where the absolute feature orientation is meaningful. Rotation equivariance and invariance can be approximated by combining the response maps of several elements of the Gabor filterbank (see Section 3.3). These response maps can be created by stepping the orientation by $\Delta\theta$ (XTGK).

The $3D$ extension of Gabor filters can be found in Qian *et al.*[42]. However, no recommendations are provided in the $3D$ case to sample scales and orientations for further feature extraction. One simple option to achieve (*i.e.* approximate) $3D$ image analysis is to extract $2D$ Gabor features as recommended above in the three orthogonal planes of the image frame of reference, followed by an averaging of the response maps over the three planes.

4.6 Wavelets

Wavelets form a large category of filtering methods based on a collection of high-pass and low-pass filters that are designed to cover the entire image spectrum[35] (see Section 3.4). Pairwise combinations of one high- and one low-pass filter result usually in a sequence of band-pass response maps (*i.e.* wavelet coefficients) with a factor of 2 between their scales and one remaining low-pass response map. Two properties must be considered when implementing wavelet transforms, namely *decimation* and *separability*. Decimation relates to the downsampling operation of the response maps and is compared in Sections 4.6.1 and 4.6.2. Separable and non-separable wavelets concern the separability of high- and low-pass filters and are detailed in Sections 4.6.3 and 4.6.4.

4.6.1 Decimated Transform PH3R

The decimated transform is not redundant and allows coding images with a minimal number of coefficients. However, the response maps containing the coefficients are iteratively decimated, which means that their size decreases throughout the levels of the decomposition, leading to a lack of translation invariance.

For instance, with separable wavelets (see Section 4.6.3), the image $f[\boldsymbol{k}]$ is first convolved with a high-pass filter g_H and a matching low-pass filter g_L along each image direction. In $2D$, this yields four response maps: h_{LL}, h_{LH}, h_{HL} and h_{HH}. All response maps are then downsampled by a factor of two in all directions to become h^1_{LL}, h^1_{LH}, h^1_{HL} and h^1_{HH}. This concludes the first iteration of the discrete wavelet transform. For the next iterations, the low-pass coefficients h^j_{LL} are subsequently convolved with g_H and g_L along each image direction and downsampled. This means that the h^j_{LL} response map of each decomposition level is used as input image for the next level $j+1$ (GCEK). It is worth noting that h^j_{LL} is traditionally discarded when the wavelet transform is used for image compression and reconstruction. Because the response maps h^j_{LL} are downsampled j times, this has the same effect as dilating (*i.e.* upsampling) the filters by 2^j in Eq. (4.6). The downsampling of the response maps yields the wavelet coefficients of iteration $j+1$. The response maps resulting from the decimated separable wavelet decomposition is illustrated in Fig. 4.9 for the $2D$ case. Although illustrated in the context of separable wavelets, decimated transforms can also be used with non-separable wavelets.

It is worth noting that the convolution is, in this case, modified because the shifts are restricted to the resolution of the j times downsampled response maps[12]. In addition, this must be taken into account when aggregating the response maps (see Section 1.3), where the ROI mask $\boldsymbol{R}$ must also be downsampled to match the dimensions of the corresponding response maps $h^j[\boldsymbol{k}]$.

4.6.2 Undecimated Transform CVCQ

The undecimated transform, also called *stationary* transform, yields a translation-invariant decomposition by obviating the downsampling steps required by the decimated transform. Although the transform becomes redundant (*i.e.* it yields more coefficients than strictly required for a perfect reconstruction), it is better suited to our case where we are not interested in image coding but rather image analysis.

The separable undecimated transform involves upsampling of the wavelet filters g^j_H and g^j_L to achieve the multiscale decomposition via the matched-size filters in Eq. (4.6). While others exist, a common upsampling approach is to use the *à trous* algorithm[21]. The *à trous* algorithm involves inserting zeros, or holes, into the filter kernel. For example, the high-pass kernel of the Haar wavelet is $\left[-1/\sqrt{2}, 1/\sqrt{2}\right]$. The first level decomposition of this high-pass kernel is $\left[-1/\sqrt{2}, 0, 1/\sqrt{2}\right]$, or

v4: Clarified implementation of the *à trous* algorithm.

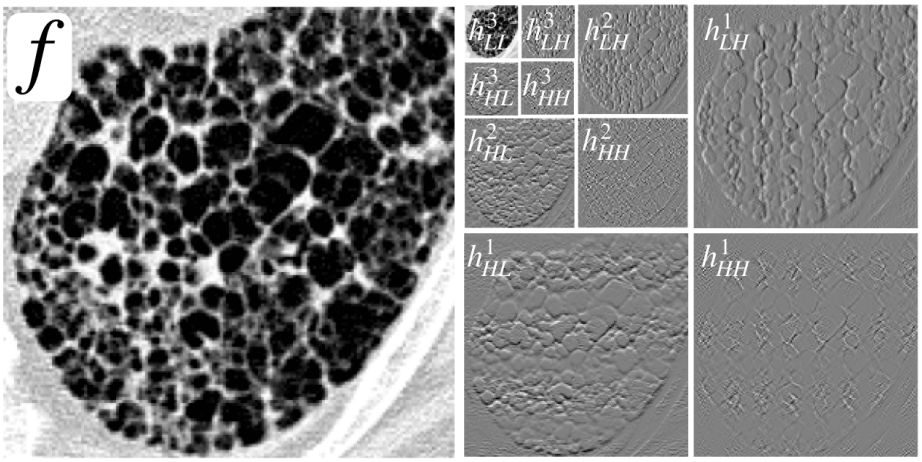


Figure 4.9 — Response maps from the $2D$ decimated separable wavelet transform of the input image f. The first three iterations are shown when using the Haar wavelet.

alternatively $\left[-1/\sqrt{2}, 0, 1/\sqrt{2}, 0\right]$. The second level decomposition of the same kernel is formed by inserting zeros between the values of the first level kernel, forming $\left[-1/\sqrt{2}, 0, 0, 0, 1/\sqrt{2}\right]$, or alternatively $\left[-1/\sqrt{2}, 0, 0, 0, 1/\sqrt{2}, 0, 0, 0\right]$. Both alternatives are valid, but result in different response maps. Popular standard implementations in `MATLAB` and `pywavelets` use the second alternative. We therefore recommend using the second alternative for reproducibility.

The response maps $h^j[\boldsymbol{k}]$ produced through undecimated transform have the same dimensionality as the input image f and are simply obtained via the convolution of $f[\boldsymbol{k}]$ with the filters g^j_H and g^j_L along each image direction. In $2D$, this yields four response maps for every iteration: h^j_{LL}, h^j_{LH}, h^j_{HL} and h^j_{HH}.

The response maps resulting from the undecimated separable wavelet decomposition is illustrated in Fig. 4.10 for the $2D$ case. Again, although illustrated here for separable wavelets, non-decimated transforms can also be used with non-separable wavelets.

4.6.3 Separable Wavelets 25BO

The discrete separable wavelet transform yields a collection of $1D$ wavelet kernels obtained from J dilations of one unique mother wavelet function, which is a high-pass filter $g_H[k]$ [12]. The remaining low frequencies are covered by a low-pass filter g_L called scaling function. When considering dyadic dilations, the scale of the kernels g_H and g_L are indexed by j as

$$g^j[k] = 2^{j/2} g[2^j k]. \tag{4.6}$$

The distinctive property of the separable wavelet transform is to use the separable convolution. The latter is computationally efficient but for radiomics has two distinct disadvantages: it is not rotationally invariant, and only strictly separable wavelets can be used (see Section 2.1).

Directionality Considerations

In $3D$, the convolution of the low-pass and high-pass filters in the three directions of space yields eight different wavelet response maps: h_{LLL}, h_{LLH}, h_{LHL}, h_{LHH}, h_{HLL}, h_{HLH}, h_{HHL} and h_{HHH}. Within the image frame of reference[d], these three directions are k_1, k_2 and k_3. For example, let us consider the h_{LHL} response map: a low-pass filter is applied in the k_1 direction,

[d] The image frame of reference of every image set (*e.g.* a scan) of a study should have the same common orientation relative to the patient reference frame, see Section 1.2.1.

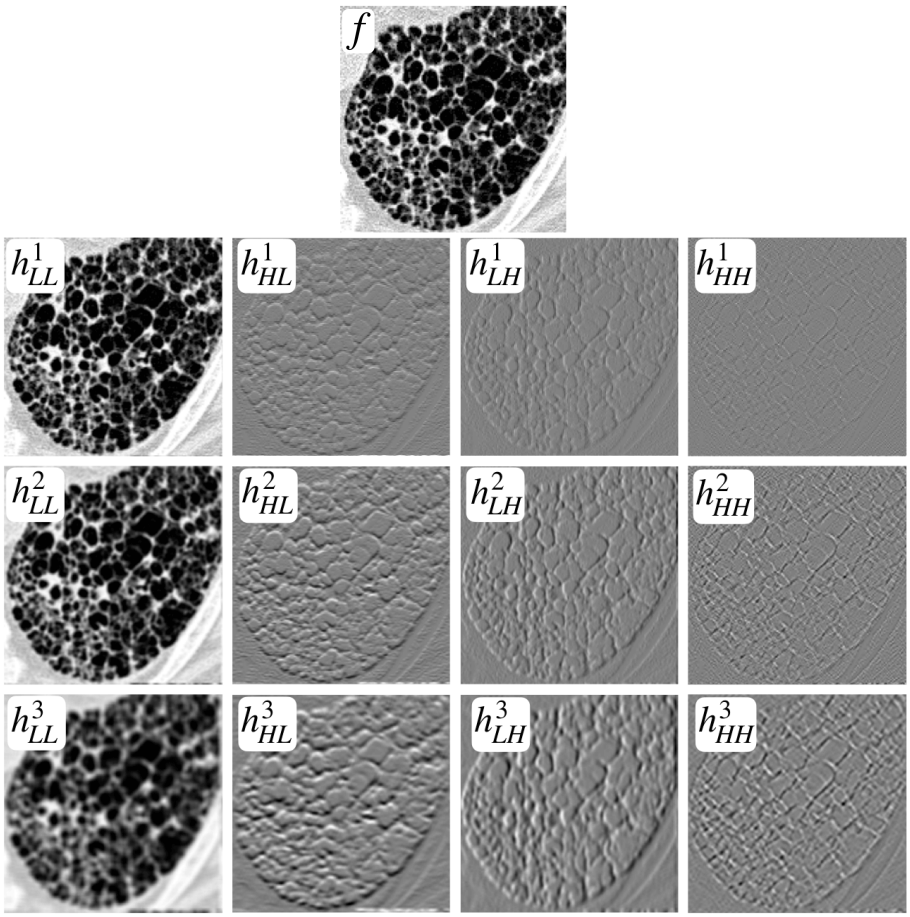


Figure 4.10 — Response maps from the $2D$ undecimated separable wavelet transform of the input image f. The first three iterations are shown when using the Haar wavelet.

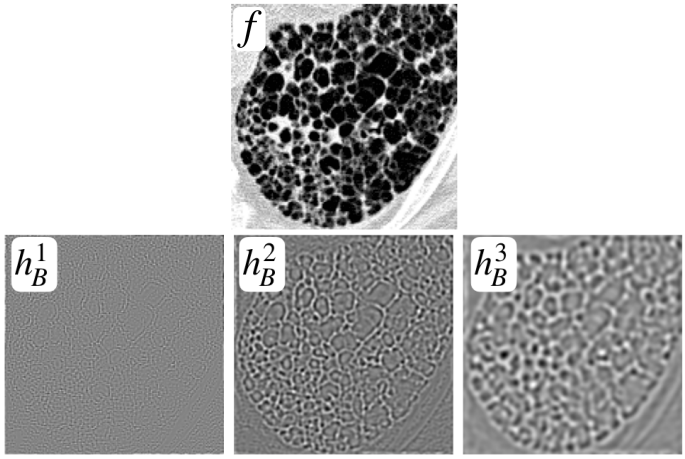


Figure 4.11 — Response maps from the $2D$ undecimated non-separable wavelet transform of the input image f. The first three iterations are shown when using the Simoncelli wavelet.

a high-pass filter is applied in the k_2 direction and a low-pass filter is applied in the k_3 direction (see Section 2.1).

Haar UOUE

The simplest separable wavelet is the Haar wavelet. The Haar wavelet and scaling function form the simplest admissible function pair and are defined as[34]

$$g_{H,\text{Haar}} = \left[\frac{-1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right], \quad g_{L,\text{Haar}} = \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right].$$

Daubechies

Daubechies wavelets and scaling functions are characterised by their number of vanishing moments p, *i.e.* their ability to approximate polynomials of order up to $p-1$[12]. For $p = 1$ Daubechies kernels are equivalent to Haar. Kernel values of wavelet and scaling functions for $p > 1$ can be obtained from the PyWavelets website[e].

Other

There exists a large number of other separable wavelet/scaling function pairs (*e.g.* Meyer, Coiflets), each targeting different objectives in terms of signal analysis. The kernel values of most common wavelets can be obtained from the Wavelet Browser of PyWavelets[f].

v6: Added implementation troubleshooting to clarify how undecimated wavelet decomposition should be implemented.

Implementation Troubleshooting

- Undecimated wavelet decomposition of separable wavelets require a sequence of steps. Here we illustrate these steps using the Haar wavelet for two iterations, where we want to compute the pseudo-rotation invariant response map of the high-pass (HHH) filter in the final iteration:
  1. iteration $j = 1$: 1^{st} decomposition level
     - Define the basic low-pass g_L kernel: $\left[1/\sqrt{2}, 1/\sqrt{2}\right]$
     - If necessary, append a zero to obtain an odd filter dimension: $g^1_{L_{odd}} = \left[1/\sqrt{2}, 1/\sqrt{2}, \mathbf{0}\right]$
     - Create a filter bank of 4 ($2D$) or 24 ($3D$) low-pass filter sets based on $g^1_{L_{odd}}$ for rotation invariance, see appendix A.
     - Apply each filter bank to obtain 4 ($2D$) or 24 ($3D$) response maps h^1_j.
  2. iteration $j = 2$: 2^{nd} decomposition level
     - Define the basic high-pass g_H kernel: $\left[-1/\sqrt{2}, 1/\sqrt{2}\right]$
     - Apply the à trous algorithm and insert zeros: $g^2_H = \left[-1/\sqrt{2}, \mathbf{0}, 1/\sqrt{2}, \mathbf{0}\right]$
     - If necessary, append a zero to obtain an odd filter dimension: $g^2_{H_{odd}} = \left[-1/\sqrt{2}, 0, 1/\sqrt{2}, 0, \mathbf{0}\right]$
     - Create a filter bank of 4 ($2D$) or 24 ($3D$) low-pass filter sets based on $g^2_{H_{odd}}$ for rotation invariance, see appendix A.
     - Apply each filter bank j to the corresponding low-pass response maps of the previous iteration (h^1_j) obtain 4 ($2D$) or 24 ($3D$) response maps h^2_j.
     - Pool the response maps, for example through averaging, to obtain response map h^2_{HHH}.

[e] http://wavelets.pybytes.com/family/db/, as of July 2018.
[f] http://wavelets.pybytes.com, as of July 2020.

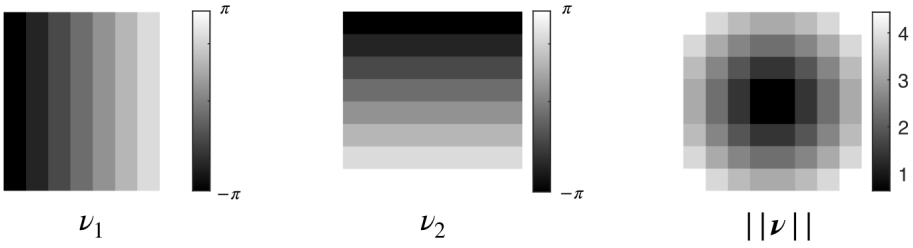


Figure 4.12 — Coordinate grids in voxel dimensions for for $N = 8$.

4.6.4 Non-separable Wavelets LODD

v4: Extended description of non-separable wavelets.

As motivated in Section 3.1, filters (and more generally texture operators) should be invariant or equivariant to local rotations in most cases. However, with the exception of the Gaussian filter, all separable filters (including separable wavelets) are not invariant/equivariant to rotations. Therefore, it is interesting to consider non-separable wavelets to achieve isotropic image analysis. The starting point for the definition of non-separable wavelets is a single (unidimensional) radial profile in the Fourier domain as $\hat{g}[\boldsymbol{\nu}]$, where $\hat{g}$ is a function of $||\boldsymbol{\nu}||$ defining the radial coordinate[48]. As a consequence, these functions are circularly symmetric and can be further combined with directional analysis such as the Riesz transform (Section 4.7).

Non-separable wavelets are easily implemented directly in the Fourier domain using their unidimensional radial profile. The support of the filter in the Fourier domain is the same as the considered $N_1 \times \cdots \times N_D$ image f to allow for efficiently computing the convolution as a simple Hadamard product via Eq. (2.3). Along dimension i, each coordinate ν_i is contained in $[-\frac{N_i}{2}, \frac{N_i}{2}]$ which is centered around the null frequency $\boldsymbol{\nu} = \mathbf{0}$. The maximum coordinate value $\frac{N_i}{2}$ will be assigned to the value $\nu_B = \pi$ radians/sample, which is called the normalized Nyquist frequency. The unidimensional radial coordinate is then obtained as $||\boldsymbol{\nu}|| = \sqrt{\nu_1^2 + \nu_2^2 + \cdots + \nu_D^2}$. It is worth noting that the value of $||\boldsymbol{\nu}||$ will exceed ν_B in the corners of the image. These "corner" values should be ignored as the corresponding frequencies can only be measured along diagonal directions, yielding anisotropic image analysis. They are ignored by construction for all proposed radial wavelets, e.g. (4.7) and (4.8).

In order to implement non-separable wavelets in practice, we recommend defining a coordinate grid in the Fourier domain, where each dimension ν_i is sampled in the interval $[-\pi, \pi]$ with a step of $\frac{2\pi}{N_i}$. Coordinate grids in voxel dimensions are illustrated in Fig. 4.12 for $N = 8$.

Whereas wavelet functions are usually defined as a cascade of high- and low-pass filters, here we focus on the band-pass filters corresponding to the consecutive subbands (i.e. response maps) that would results from combined high- and low- pass filtering (see introductory paragraph of Section 4.6). In other words, these response maps can be directly obtained using a corresponding band-pass filter $h_B[\boldsymbol{k}]$, without the need of re-convolving with previous iterations of the wavelet transform. The resulting response maps are referred to as "B maps".

The simplest non-separable and circularly symmetric wavelet is the Shannon wavelet[48] (GWM2) characterised by the following mother function

$$\hat{g}_{\text{sha}}[\boldsymbol{\nu}] = \begin{cases} 1 & \text{if} \quad \frac{\nu_B}{2} < ||\boldsymbol{\nu}|| \leq \nu_B, \\ 0 & \text{otherwise.} \end{cases} \tag{4.7}$$

Eq. (4.7) corresponds to the sinc wavelet in the spatial domain, which has the disadvantage of having large spatial supports. Therefore, an interesting alternative is the smoother and more

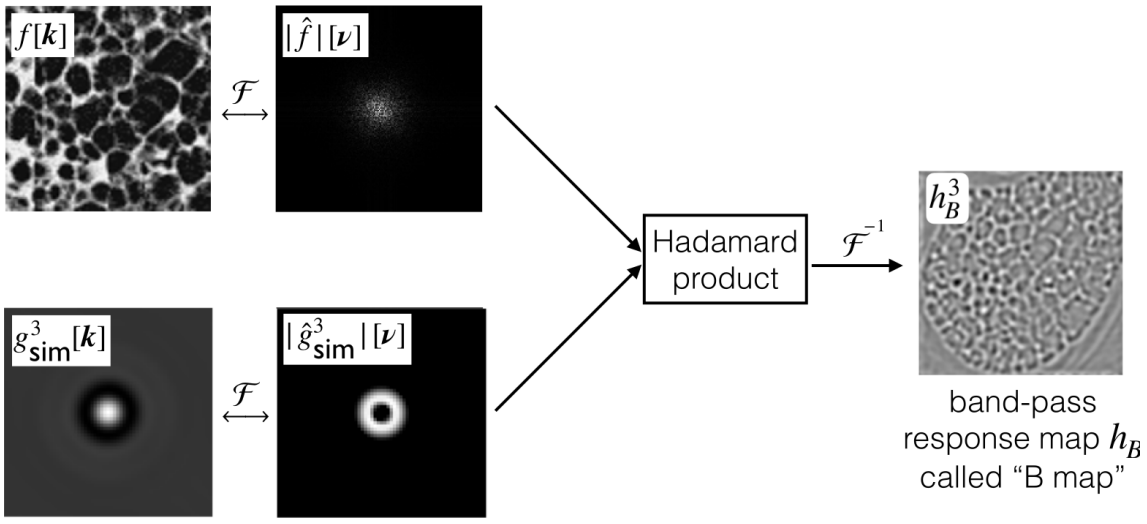


Figure 4.13 — Convolution in the Fourier domain using a Hadamard product. Complex modulus of the Fourier images are shown only for display purposes: the Hadamard product must be done with complex values.

Figure 4.14 — Example of a $2D$ band-pass Simoncelli wavelet g_{sim} after three decomposition levels, computed on a 66×66 grid. The wavelet is shown in the spatial (left) and in the Fourier (right, complex modulus shown) domains.

compactly supported (in space) Simoncelli wavelet[40] (PRT7). Its band-pass function is defined as

$$\hat{g}_{\text{sim}}[\boldsymbol{\nu}] = \begin{cases} \cos\left(\frac{\pi}{2}\log_2\left(\frac{2||\boldsymbol{\nu}||}{\nu_B}\right)\right) & \text{if} \quad \frac{\nu_B}{4} \leq ||\boldsymbol{\nu}|| \leq \nu_B, \\ 0 \quad \text{otherwise.} \end{cases} \tag{4.8}$$

An easy way to implement several consecutive iterations of such a non-separable wavelet transform is to start from the initial definition of the band-pass filter (the one with highest frequency band, *e.g.* Eq. (4.8)), and constructing the filter with half ν_B at every consecutive step. Concretely, the bandpass filter at a given scale level j is obtained by replacing ν_B by $\frac{\nu_B}{2^j}$ in (4.7) or (4.8). The response map can then be computed by taking the Hadamard product of the bandpass filter and the image in Fourier domain, and then performing an inverse Fourier transform. This process is illustrated in Fig. 4.13.

The response maps resulting from the undecimated non-separable wavelet decomposition is illustrated in Fig. 4.11 for the $2D$ case. An example of a Simoncelli wavelet is shown in Fig. 4.14. Other alternatives can be found in Table 1 of Unser *et al.*[48].

4.6.5 Wavelets: Considerations for Radiomics

To summarise, undecimated non-separable wavelets have the advantage of yielding isotropic (*i.e.* rotation invariant/equivariant, see Section 3.3) and translation equivariant image analysis. Moreover, they yield only one response per decomposition level (*i.e.* one per scale), which significantly reduces the number of radiomics features when compared to their separable counterpart. Separable wavelets were mostly designed for image coding, which has very different design constraints. Such wavelets yield a large collection of response maps that are biased towards image axes and lack rotational invariance. While it is possible to make them approximately rotation invariant using orientation pooling over equivariant right angle representations as suggested in Section 3.3 and Appendix A, we do not recommend to use this orientation pooling procedure for decomposition levels larger than 1. Applying orientation pooling after every convolution operation of a multi-level separable wavelet decomposition would result in an overly complicated algorithm, yielding only approximate rotational invariance where the granularity of the approximation depends on the number of group elements (*e.g.* right angles only or using more angular samples) in the used equivariant representation. Therefore, we recommend not to use separable wavelet for rotation invariant image analysis, or to limit the wavelet decomposition to only one level. Isotropic non-separable wavelets such as the Simoncelli wavelet allow achieving multi-level, truly isotropic and rotation invariant image analysis by design. When the directionality of image patterns of interest (*e.g.*, tumour margin, vessels) is expected to be important, aligned directional wavelet filters such as Riesz (Section 4.7) that allow combining directional sensitivity with local invariance to rotations can be considered (see Section 3.3).

v4: Added recommendation to not perform decomposition using separable wavelets beyond the first level.

4.7 Riesz AYRS

Whereas non-separable wavelets like Simoncelli constitute an excellent first solution for using wavelets in radiomics studies, they are also circularly symmetric and therefore cannot characterise directional patterns. To address directionality of patterns, these non-separable circularly symmetric wavelets can be advantageously combined with directional analysis methods such as directional derivatives[41] or spherical harmonics.

v4: Extended description of the Riesz transform.

An elegant approach to obtain features measuring directional transitions between pixel values is to use image derivatives. Besides being simple to compute, these have the advantage of being interpretable (at least for the first- and second-order), which makes them attractive for understanding the meaning of the texture measures in a particular medical or biological application context. For instance, the first-order derivative is called the gradient and informs about the slope of the transitions in the image, *e.g.* sharp versus smooth transitions. Second-order derivatives are called Hessian and quantify the curvature of the transitions. It is worth noting that the spatial scale of these transitions will be controlled by a preliminary isotropic filtering step based on *e.g.* a Gaussian smoother (low-pass) or a Simoncelli wavelet (band-pass).

An interesting option to compute the derivatives is to do so in the Fourier domain[14]. This also provides the opportunity to easily compute higher-order image derivatives of order l (0C48) as

$$\frac{\partial^l}{\partial k_i^l} f[\boldsymbol{k}] \overset{\mathcal{F}}{\longleftrightarrow} (\,\mathrm{j}\nu_i\,)^l \hat{f}[\boldsymbol{\nu}], \tag{4.9}$$

where $1 \leq i \leq D$. It can be noticed that differentiating an image along the direction k_i only requires multiplying its Fourier transform by $\mathrm{j}\nu_i$. Computing l^{th}-order derivatives has an intuitive interpretation (*e.g.*, gradient for $l = 1$, curvature for $l = 2$). Let us illustrate this by differentiating a simple $2D$ Gaussian function of dimension 32×32 (see Fig. 4.15). A Gaussian function $f[\boldsymbol{k}]$

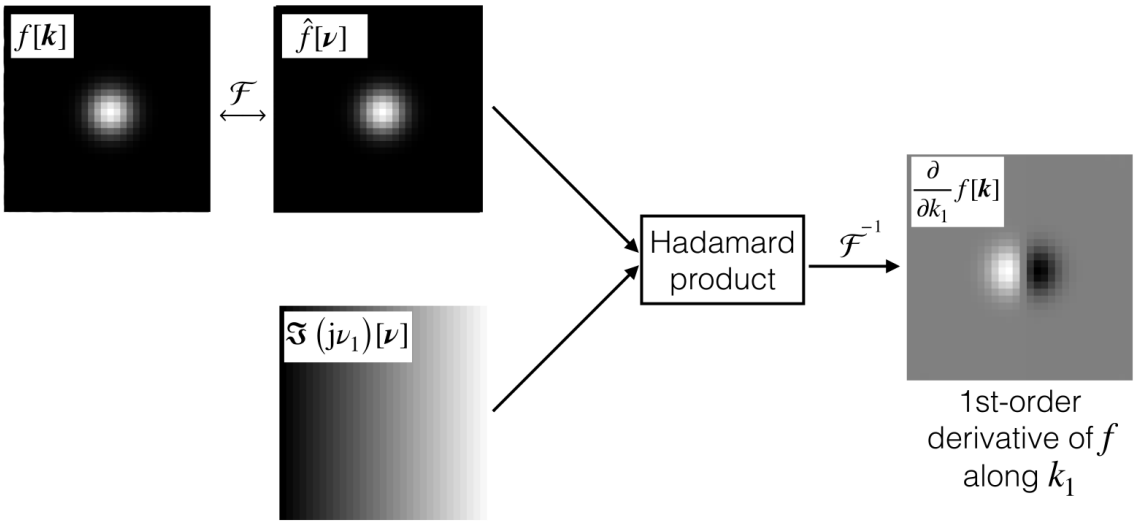


Figure 4.15 — Deriving an image in the Fourier domain (illustrated with a Gaussian function). The notation $\Im(\cdot)$ denotes the imaginary part.

is first transformed in the Fourier domain resulting in $\hat{f}[\boldsymbol{\nu}]$. It is worth noting that the Fourier transform of a Gaussian is real-valued and also a Gaussian. In parallel, the derivative kernel $\mathrm{j}\nu_1$ can be simply defined by multiplying the Fourier coordinate grid ν_1 (see Fig. 4.12) by the imaginary unit j. Whe using the Hadamard product in Fourier and then back to the space domain (*i.e.* convolution operation as depicted in Fig. 4.13), the result corresponds to the derivative of f along k_1, $\frac{\partial}{\partial k_1} f[\boldsymbol{k}]$.

Unfortunately, a pure image derivative filter as computed in Eq. (4.9) is high-pass (because multiplied by ν_i) and accentuates high frequencies along k_i. Therefore, it is desirable to implement image derivatives as all-pass filters instead of high-pass, by simply normalising the derivative kernel $\mathrm{j}\nu_i$ by the coordinate norm $\boldsymbol{\nu}$. This is exactly what the first-order Riesz transform yields $\boldsymbol{\mathcal{R}}\{f\}[\boldsymbol{k}]$ as[47]

$$\boldsymbol{\mathcal{R}}\{f\}[\boldsymbol{k}] = \begin{pmatrix} \mathcal{R}_1\{f\}[\boldsymbol{k}] \\ \vdots \\ \mathcal{R}_D\{f\}[\boldsymbol{k}] \end{pmatrix} \overset{\mathcal{F}}{\longleftrightarrow} -\mathrm{j}\frac{\boldsymbol{\nu}}{||\boldsymbol{\nu}||}\hat{f}[\boldsymbol{\nu}]. \tag{4.10}$$

It can be noticed that dividing the Fourier representation by the norm of $\boldsymbol{\nu}$ transforms Eq. (4.9) in D all-pass operators $\mathcal{R}_i$. Eq. (4.10) can be used to compute first-order directional derivatives (*i.e.* gradient-like for characterising image edges). However, higher-order derivatives can be relevant for radiomics studies (*e.g.* second-order or Hessian-like for characterising image ridges like tumour margin or vessels).

A qualitative comparison between classical image derivatives as defined in (4.9) and Riesz-based image derivatives (4.10) is proposed in Fig. 4.16 for both a Gaussian image (f_1) and a white noise image (f_2).

For a fixed (maximal) order L, the collection of higher-order all-pass image derivatives are defined in the Fourier domain as

$$\widehat{\boldsymbol{\mathcal{R}}}^{\boldsymbol{l}}\{\hat{f}\}[\boldsymbol{\nu}] = (-\mathrm{j})^L \sqrt{\frac{L!}{l_1!\cdots l_D!}} \frac{\nu_1^{l_1}\cdots\nu_D^{l_D}}{(\nu_1^2+\cdots+\nu_D^2)^{L/2}}\hat{f}[\boldsymbol{\nu}], \tag{4.11}$$

which yields a total of $\binom{L+D-1}{D-1} = \frac{(L+D-1)!}{L!(D-1)!}$ all-pass filters for all combinations of the elements l_i of the vector $\boldsymbol{l}$ as $|\boldsymbol{l}| = l_1+\cdots+l_D = L$. The collection of Riesz operators $\boldsymbol{\mathcal{R}}^{(l_1,l_2,\ldots,l_D)}$ of order

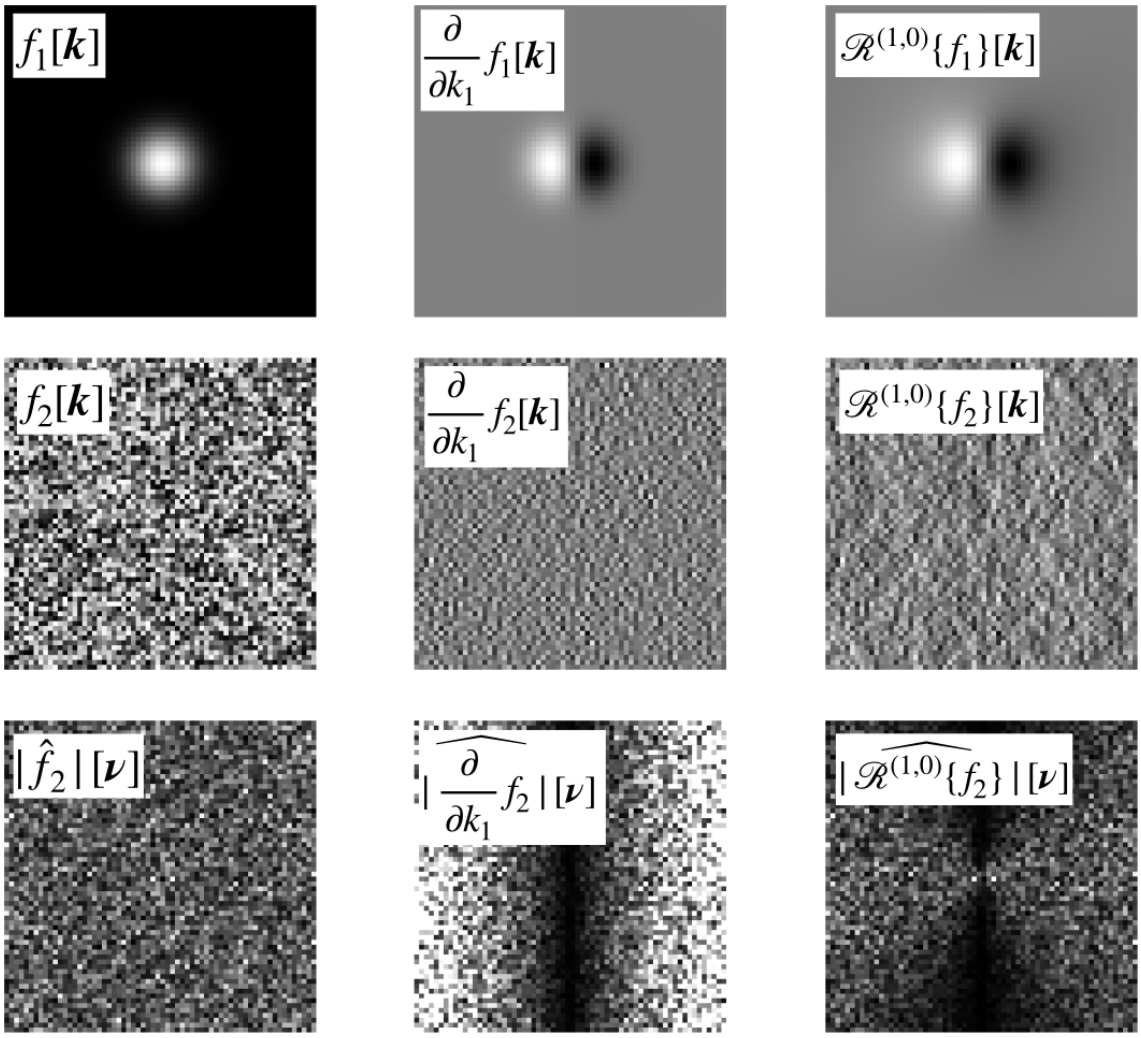
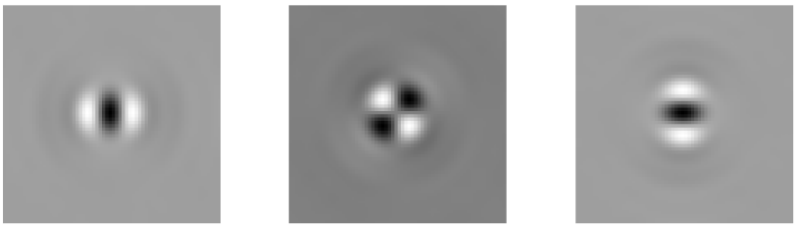


Figure 4.16 — Qualitative comparison between classical image derivatives (high-pass) as defined in (4.9) and Riesz-based image derivatives (all-pass) as defined in (4.10).
First row: Gaussian image f_1 and its derivatives in the spatial domain. The classical derivative $\frac{\partial}{\partial k_1} f_1[\boldsymbol{k}]$ and the Riesz transform $\mathcal{R}^{(1,0)}\{f_1\}[\boldsymbol{k}]$ look very similar because the Gaussian image is essentially composed of low frequencies.
Second row: White noise image f_2 and its derivatives in the spatial domain. The classical derivative $\frac{\partial}{\partial k_1} f_2[\boldsymbol{k}]$ contains mostly high frequencies, whereas the Riesz derivative $\mathcal{R}^{(1,0)}\{f_2\}[\boldsymbol{k}]$ retains all frequencies.
Third row: White noise image $\hat{f}_2$ and its derivatives in the Fourier domain (moduluses are shown). The difference between the two derivatives approaches is striking, where the boost of high frequencies along ν_1 for the classical derivative approach can be clearly seen in $|\widehat{\frac{\partial}{\partial k_1} f_2}|[\boldsymbol{\nu}]$. The vertical and centered black line results from the fact that first order derivatives are zero mean (*i.e.* they contain no null frequencies).



$\mathcal{R}^{(2,0)}\{g_{\text{sim}}\}[\boldsymbol{k}]$ $\mathcal{R}^{(1,1)}\{g_{\text{sim}}\}[\boldsymbol{k}]$ $\mathcal{R}^{(0,2)}\{g_{\text{sim}}\}[\boldsymbol{k}]$

Figure 4.17 — A $2D$ Riesz filterbank for $L = 2$ and combined with a Simoncelli wavelet.

L is denoted by $\boldsymbol{\mathcal{R}}^L$. For instance, in $3D$ and with $L = 2$, the element $\widehat{\mathcal{R}}^{(0,2,0)}\{f\}[\boldsymbol{\nu}]$ corresponds qualitatively to a second-order derivative of f along the direction k_2 and we have

$$\widehat{\mathcal{R}}^{(0,2,0)}\{f\}[\boldsymbol{\nu}] = \frac{-\nu_2^2}{\nu_1^2 + \nu_2^2 + \nu_3^2}\,\hat{f}[\boldsymbol{\nu}].$$

A set of band-pass, multi-scale and multi-orientation filters $\boldsymbol{g}_{\sigma,\boldsymbol{l}}[\boldsymbol{k}]$ can be obtained by simply applying the Riesz transform to circularly symmetric non-separable wavelets *e.g.* Eq. (4.8) or multi-scale filters *e.g.* the LoG filter g_σ Eq. (4.2), as

$$\boldsymbol{g}_{\sigma,\boldsymbol{l}}[\boldsymbol{k}] = \boldsymbol{\mathcal{R}}^{\boldsymbol{l}}\{g_\sigma\}[\boldsymbol{k}]. \tag{4.12}$$

An example of a $2D$ Riesz filterbank for $L = 2$ and combined with a Simoncelli wavelet is shown in Fig 4.17. Eq. (4.12) yields a collection of filters measuring L-th order (scaled) derivatives. However, these derivatives are computed along image axes $k_1, \ldots, k_D$, which entails similar challenges as separable wavelets: (i) analyzing images along their axes does not have a particular signification for the problem at hand (*e.g.* characterising local tissue structures within a tumour) and (ii) they are not rotation invariant/equivariant.

4.7.1 Aligning Riesz kernels 1SD3

v6: Additional clarifications on how to align Riesz kernels were provided.

To address (i) and (ii), one can locally align all Riesz filters using a given alignment criterion to combine directional sensitivity with local invariance to rotations, as motivated in Section 3.3 and Fig. 3.1. Whereas the max orientation pooling operation can be seen as an alignment criterion, the latter only focuses on the maximum response of one single filter. Since Riesz filter banks include several filters with distinct profiles (see *e.g.* Fig. 4.17 with order $L = 2$), other alignment criteria based on meaningful quantities can be used. We will seek for alignment criteria that are optimizing the response of the entire filterbank and not only one single kernel.

Two suitable alignment criteria are those maximizing the gradients or the Hessian[19], which can be efficiently computed directly from the Riesz coefficients using the structure (or Hessian) tensor $\mathrm{J}[\boldsymbol{k}_0]$[8]. It is worth noting that the while classical definition of the structure tensor is based on simple (high-pass) image derivatives, we can use in this context the all-pass image derivatives provided by the Riesz transform. In practice, it is beneficial to regularise this structure tensor using a Gaussian window $g_{\sigma_{\text{tensor}}}[\boldsymbol{k}]$ (noted as $\nu(\boldsymbol{x})$ in Section IV-B of[8]). Regularizing the structure tensor is important to determine the scale of the structures from which the orientation is important at a given position $\boldsymbol{k}_0$. This must be optimized for the problem at hand via the variance σ_{tensor} of a Gaussian window $g_{\sigma_{\text{tensor}}}[\boldsymbol{k}]$. It is worth noting that $g_{\sigma_{\text{tensor}}}$ is independent from the multi-scale framework used to compute the Riesz transform (e.g. Simoncelli or the LoG filter, see Eq. (4.12)).

In $3D$, the nine elements of the regularised structure tensor computed from Riesz coefficients

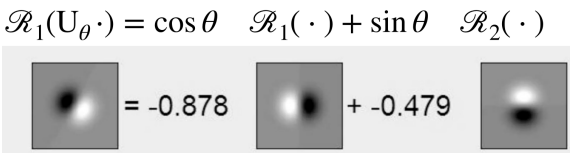


Figure 4.18 — Illustration of the steerability property of the Riesz transform in $2D$. Rotating the Riesz kernel $\mathcal{R}_1$ using a rotation matrix U_θ only requires computing a linear combination[47].

at the position $\boldsymbol{k}_0$ are defined using (see Section IV-B of Chenouard *et al.*[8])

$$\Big[\mathrm{J}[\boldsymbol{k}_0]\Big]_{mn} = \sum_{\boldsymbol{k}\in\mathbb{Z}^D} g_{\sigma_{\text{tensor}}}[\boldsymbol{k}-\boldsymbol{k}_0]\cdot\mathcal{R}_m\{f\}[\boldsymbol{k}]\cdot\mathcal{R}_n\{f\}[\boldsymbol{k}], \tag{4.13}$$

where the indices $m, n \in \{1,2,3\}$ identify the combinations of all first-order $3D$ Riesz components[g]. It is worth noting that while the sum is defined over the entire image domain $N_1\times\cdots\times N_D \subset \mathbb{Z}^D$, a smaller window can be considered in practice based on the decay of the Gaussian window $g_{\sigma_{\text{tensor}}}[\boldsymbol{k}]$. The collection of eigenvectors $(\boldsymbol{u}_1,\boldsymbol{u}_2,\boldsymbol{u}_3)$ of J, sorted by eigenvalues, defines a rotation matrix U for every position $\boldsymbol{k}_0$. In $3D$ for instance, U is a 3×3 rotation matrix that maximizes the energy of $\mathcal{R}_1$, then maximizes the residual energy of $\mathcal{R}_2$, and then $\mathcal{R}_3$. Therefore, U constitutes an interesting alignment criteria for the position $\boldsymbol{k}_0$. Computing J and then U for every position in the image defines alignment maps, which however come with a high computational cost.

Computing these alignment maps allows determining how to further orient all elements of the Riesz filter bank to achieve locally rotation invariant image analysis. Thanks to the steerability property of the Riesz transform[47], no additional convolution operations are required to compute the response of rotated filters. A steerable filter[26] is a filter that can be obtained as a linear combination of steerable kernels, and all rotations of the former can also be obtained via another linear combination of the filter responses parameterised by the desired rotation (*e.g.* one rotation angle in $2D$), which obviates the need to re-convolve the oriented filter with the input image. In $2D$, steering the kernel $\mathcal{R}_1$ using a rotation matrix $\mathrm{U}_\theta = \begin{pmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{pmatrix}$ can simply be obtained using $\mathcal{R}_1(\mathrm{U}_\theta\,\cdot) = \cos\theta\,\mathcal{R}_1(\cdot) + \sin\theta\,\mathcal{R}_2(\cdot)$, which only requires computing a linear combination parameterised by the rotation. This is illustrated in Fig. 4.18.

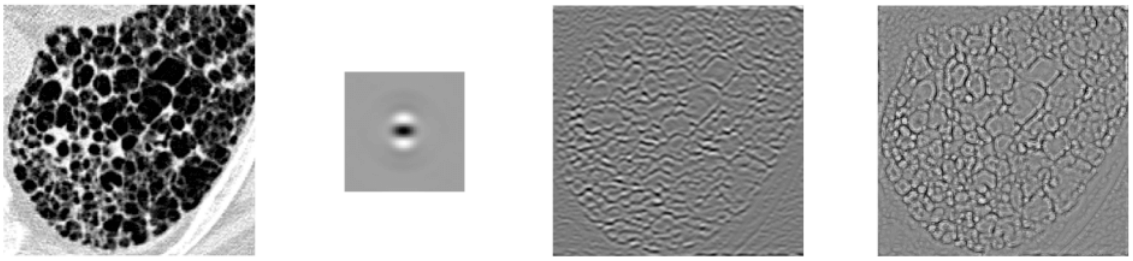
More generally, rotating all elements of the Riesz transform can be obtained with the linear combination

$$\boldsymbol{\mathcal{R}}\{f_{\mathrm{U}}\} = \mathrm{S}_{\mathrm{U}}\boldsymbol{\mathcal{R}}\{f\}, \tag{4.14}$$

where S_{U} is called the steering matrix corresponding to the rotation matrix U. S_{U} is a square matrix with a number of row/columns equal to the number of elements in the Riesz filterbank, see Eq.(4.11). Let us detail the form of S_{U} in the $3D$ case, for a rotation matrix $\mathrm{U} = (\boldsymbol{u}_1,\boldsymbol{u}_2,\boldsymbol{u}_3)^T$ with $\boldsymbol{u}_i \in \mathbb{R}^3$. For the $2D$ case, we refer to the Section II-E of Unser et al.[47]. To specifically define the elements of this steering matrix in the $3D$ case, let us start with the introduction of the multi-index notation as used by Chenouard et al.[8]. We consider index vectors of the form $\boldsymbol{n} = (n_1,n_2,n_3) \in \mathbb{N}^3$. The following multi–index notations and operators are used:

- Sum of components: $|\boldsymbol{n}| = n_1+n_2+n_3$,
- Max of components: $\max(\boldsymbol{n}) = \max(n_1,n_2,n_3)$,
- Factorial: $\boldsymbol{n}! = n_1!n_2!n_3!$,

[g] To simplify the notations, the element e.g. $\mathcal{R}^{(1,0,0)}$ is noted $\mathcal{R}_1$, the element $\mathcal{R}^{(0,1,0)}$ is noted $\mathcal{R}_2$, etc.



$f[\boldsymbol{k}]$ $\mathcal{R}^{(0,2)}\{g_{\text{sim}}\}[\boldsymbol{k}]$ $h[\boldsymbol{k}]$ $h_{\text{aligned}}[\boldsymbol{k}]$

Figure 4.19 — Example of image filtering in $2D$ with the Riesz transform. $h_{\text{aligned}}[\boldsymbol{k}]$ was obtained by aligning the responses of $\mathcal{R}^{(0,2)}\{g_{\text{sim}}\}[\boldsymbol{k}]$ based on the structure tensor and allows highlighting of collagen fibers at any orientation.

- Exponentiation of a vector $\boldsymbol{x} = (x_1, x_2, x_3) \in \mathbb{R}^3$: $\boldsymbol{x}^{\boldsymbol{n}} = x_1^{n_1} x_2^{n_2} x_3^{n_3}$.

Using this multi-index notation, we can define the steering coefficients as elements $s_{\boldsymbol{n},\boldsymbol{m}}$ of the steering matrix $\mathrm{S_U}$ in the following compact form (Section III-E of[8]):

$$s_{\boldsymbol{n},\boldsymbol{m}} = \sqrt{\frac{\boldsymbol{m}!}{\boldsymbol{n}!}} \sum_{|\boldsymbol{v}_1|=n_1} \sum_{|\boldsymbol{v}_2|=n_2} \sum_{|\boldsymbol{v}_3|=n_3} \delta_{\boldsymbol{v}_1+\boldsymbol{v}_2+\boldsymbol{v}_3,\boldsymbol{m}} \cdot \frac{\boldsymbol{n}!}{\boldsymbol{v}_1!\boldsymbol{v}_2!\boldsymbol{v}_3!} \boldsymbol{u}_1^{\boldsymbol{v}_1} \boldsymbol{u}_2^{\boldsymbol{v}_2} \boldsymbol{u}_3^{\boldsymbol{v}_3},$$

where $\boldsymbol{v}_i \in \mathbb{N}^3$ and $\delta_{\boldsymbol{v}_1+\boldsymbol{v}_2+\boldsymbol{v}_3,\boldsymbol{m}}$ is the Kronecker symbol used to exclude the summation terms with $\boldsymbol{v}_1 + \boldsymbol{v}_2 + \boldsymbol{v}_3 \neq \boldsymbol{m}$.

Applying (4.14) at every image position will align the entire filterbank and yield aligned response maps $h_{\text{aligned}}[\boldsymbol{k}]$ allowing both directional and rotation-invariant image analysis. An example of image filtering in $2D$ with the Riesz kernel $\mathcal{R}^{(0,2)}\{g_{\text{sim}}\}[\boldsymbol{k}]$ is shown in Fig. 4.19. It is worth noting that although being slightly sub-optimal from a computational aspect, bypassing steerability is possible. In that case, one need to rotate the Riesz kernels using U and interpolation, and then to (re-)convolve the image with the rotated kernels to obtain the aligned response maps $h_{\text{aligned}}[\boldsymbol{k}]$.

To summarise, aligned Riesz filters allow directional and rotation-invariant image analysis that can characterise interpretable transitions in medical images. Rotationally-invariant steerable representations can also be learned to be specific to the problem at hand[1,16], also with CNNs[1,2,5,52,53,55]. Implementations of the Riesz transform are available, e.g. in[29(h)],[8(i)] and its adaptation for texture feature extraction, including filter alignment, in[19(j)].

4.8 Qualitative Comparison of Linear Filter Properties

Based on the quantitative comparison criteria introduced in Section 3, it appears that not all approaches can fulfill all properties that are desirable for local medical image analysis. One typically observes a trade off between implementation simplicity and efficiency versus filter specificity and invariance/equivariance to geometric transforms[17]. Table 4.1 provides a qualitative comparison of the filtering methods detailed in Section 4.

While approximated rotation invariance can be artificially added to all methods by computing the average of directional response maps, this often annihilates directional sensitivity (see Section 3.3). Other designs allow combining rotation invariance with directional sensitivity (see Section 3.3).

[h] `https://pypi.org/project/itk-isotropicwavelets/`, as of July 2019.
[i] `http://bigwww.epfl.ch/demo/steerable-wavelets-3d/`, as of July 2019.
[j] `http://publications.hevs.ch/index.php/publications/show/2035/`, as of July 2019.

Table 4.1 — Qualitative comparison of common linear filtering approaches. "yes/no" indicates filters that can be but are not necessarily designed as wavelets.

	directional sensitivity	local rotation invariance	separability of the convolution	wavelet
LoG	no	yes	no (yes with DoG)	yes/no
Laws kernels	yes	no	yes	no
Gabor	yes	no	no	yes/no
separable wavelets	yes	no	yes	yes
isotropic non-separable wavelets	no	yes	no	yes
aligned wavelets (*e.g.* Riesz)	yes	yes	no	yes

4.9 Interpolation and convolutional filtering

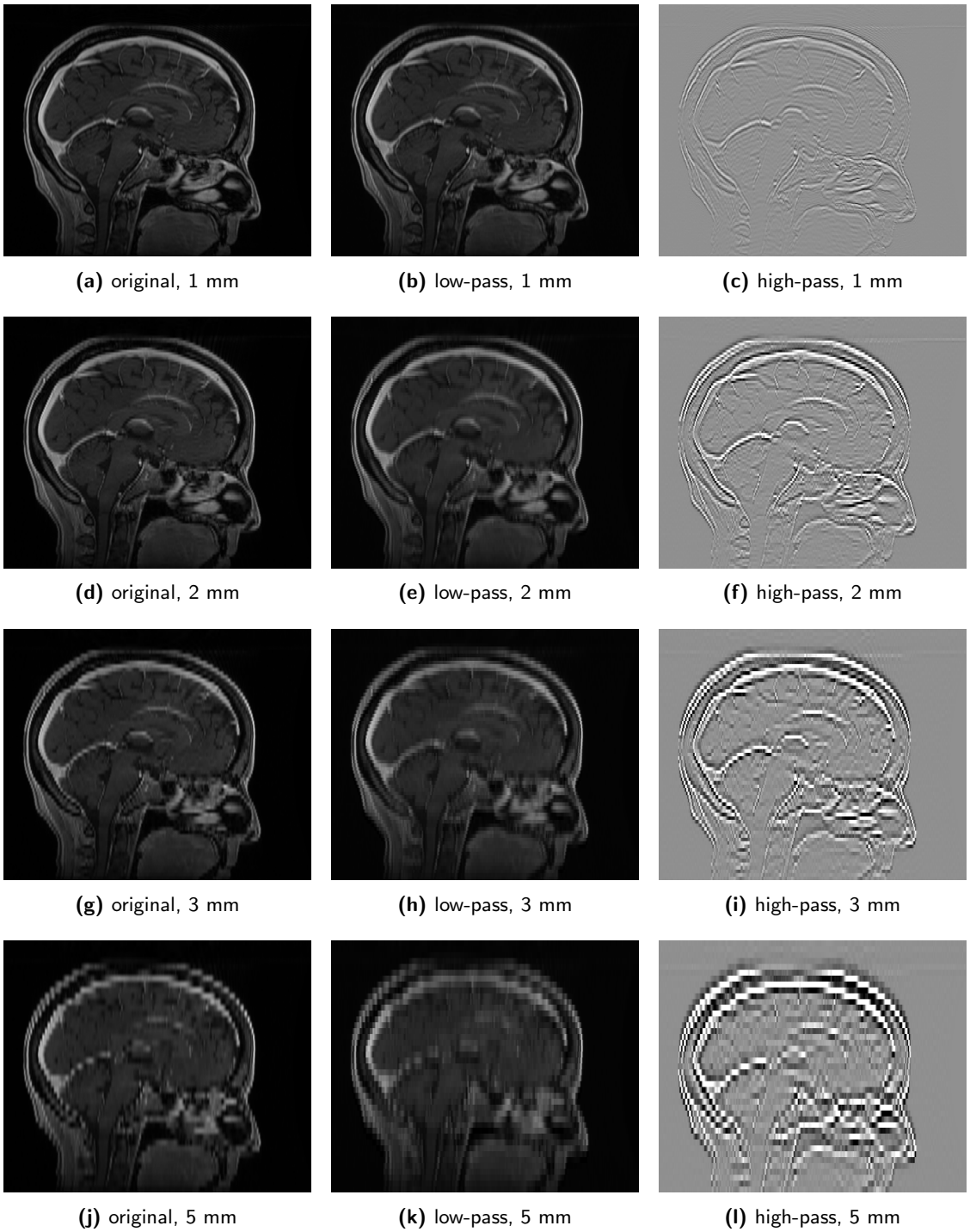
When applied to an image, displacements of one voxel may not correspond to a displacement of the same physical distance depending on the direction. Physical distance is defined by voxel spacing which, in $3D$ imaging, commonly differs between directions and images. This case is referred to as "anisotropic" image resolution. For example, in-plane voxel spacings in computed tomography (CT) images are usually smaller than the slice thickness. Also, voxel spacing can differ between acquisition protocols. Thus, simply applying filters to all images generates response maps that cannot be directly compared, because the frequency response of filters is different. To avoid this problem, we recommend performing filtering after resampling the image to uniform voxel spacing using interpolation (see Fig. 1.1).

However, performing image interpolation prior to filtering yields its own issues (see Fig. 4.20). Interpolation itself, by definition, alters the frequency content of the image, which may be particularly visible in response maps created by high-pass filters (see Fig. 4.21). As can be observed from the figure, trilinear interpolation acts as a low-pass filter when upsampling, whereas the tricubic spline method keeps preserves more high-frequency content. Ideal interpolation methods for upsampling do not exist as it is impossible to completely infer missing data. While we expect that superresolution based on deep learning could be used to improve interpolation results in the future[20], in the meantime features derived from response maps created by high-pass filters may lack robustness.

Alternatively, when an analytical expression of the filter is available, the latter is defined in the continuous domain, for example for a Gaussian filter. This provides the opportunity to (re)compute filter values for anisotropic voxel grids as well as inter-image/patient differences in image resolution and obviates interpolation. However, for clarity, we will not consider this approach in the benchmarking process (Section 6).

For most applications, boundary conditions are not critical as ROIs do not tend to be located at an image boundary. Thus, for many medical images, the nearest value boundary conditions may be used. If the ROI is close to a boundary, mirror boundary conditions may be preferable (see Section 2.2).

In general, image features, or a subset of features, may be calculated from response maps in the same way as for the original image. Image filtering also affects the discretisation method that can be used prior to computing, for example texture features. Intensities in most response maps no longer have a direct physical meaning. Hence, *fixed bin size* methods can no longer be used for response maps. *Fixed bin number* or similar discretisation methods should be used instead[58].



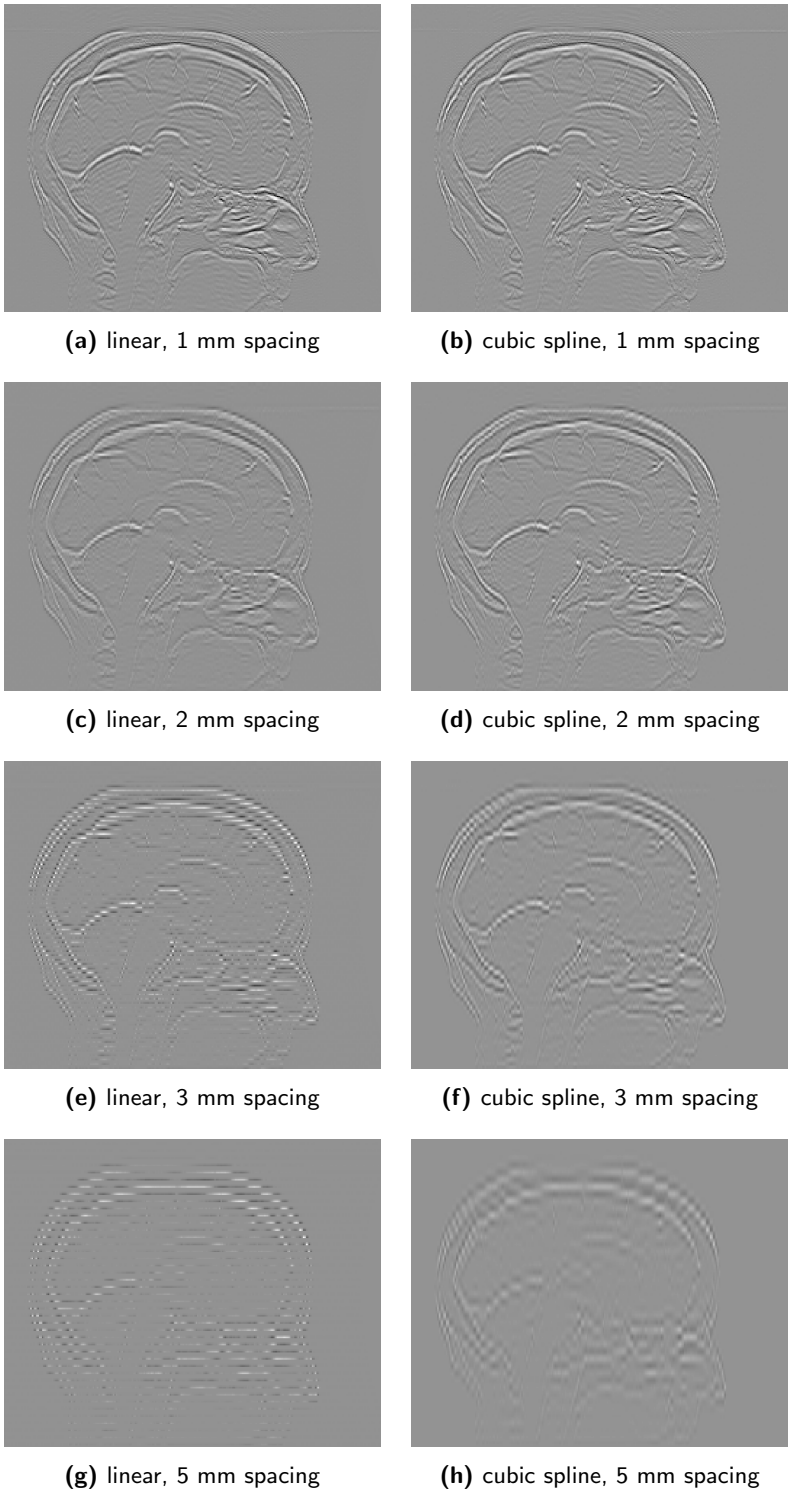
(a) original, 1 mm **(b)** low-pass, 1 mm **(c)** high-pass, 1 mm

(d) original, 2 mm **(e)** low-pass, 2 mm **(f)** high-pass, 2 mm

(g) original, 3 mm **(h)** low-pass, 3 mm **(i)** high-pass, 3 mm

(j) original, 5 mm **(k)** low-pass, 5 mm **(l)** high-pass, 5 mm

Figure 4.20 — Comparison of the low- and high-pass filters (LLL and HHH, respectively) of the Daubechies 2 wavelet for different slice spacing, without interpolation prior to filtering. All original images, all low-pass and all high-pass response maps share the same intensity scale. Notice that the high-pass filter response visibly depends on the slice spacing. This effect is also present in the low-pass image, but less so. The T1-weighted spoiled gradient echo image dataset was generated by the National Cancer Institute Clinical Proteomic Tumor Analysis Consortium (CPTAC)[9,37].



(a) linear, 1 mm spacing **(b)** cubic spline, 1 mm spacing

(c) linear, 2 mm spacing **(d)** cubic spline, 2 mm spacing

(e) linear, 3 mm spacing **(f)** cubic spline, 3 mm spacing

(g) linear, 5 mm spacing **(h)** cubic spline, 5 mm spacing

Figure 4.21 — Comparison of response maps of the Daubechies 2 wavelet high-pass filter (HHH) for images with a different slice spacing that are resampled to 1 mm isotropic voxels using trilinear or tricubic spline interpolation prior to filtering. Images with the same interpolation method have the same intensity scale. Unlike in Figure 4.20, image intensities remain similar, though the intensity range degrades with greater slice spacing. The response maps of 3 mm and 5 mm images obtained after trilinear interpolation show clear intermittent gaps between locations of the original slice. Within these slices, linear interpolation acts as a low-pass filter that suppresses high-frequency content. The cubic spline interpolation does not suffer from this problem. The T1-weighted spoiled gradient echo image dataset ($0.5 \times 0.5 \times 1.0$ mm) was generated by the National Cancer Institute Clinical Proteomic Tumor Analysis Consortium (CPTAC)[9,37].

5.2 Detailed reporting on convolutional filters

This reference manual describes parameters for a number of convolutional filters.

5.2.1 General parameters

The parameters listed below apply to every filter.

parameter	value	id
filter application		4AGS
	within the axial plane (2D)	7QSE
	volumetric (3D)	0ODV
boundary condition		GBYQ
	constant value (#)	Z3VE
	nearest value	SIJG
	periodic	Z7YO
	mirror	ZDTV

Table 5.2 — General filter parameters. Constant value padding requires the value to be specified.

5.2.2 Mean filter **S60F**

The parameters listed below apply to the mean filter.

parameter	value	id
support	$M = \#$	YNOF

Table 5.3 — Filter parameter for the mean filter.

5.2.3 Gaussian filter 8BC3

The parameters listed below apply to the Gaussian filter.

parameter	value	id
scale	$\sigma^* = \#$	41LN
filter size cutoff	#	WGPM

Table 5.4 — Parameters for Gaussian filters. Scale is expressed in physical dimensions, e.g. mm. The filter size cutoff is usually expressed in terms of the scale, e.g. $4\sigma^*$.

5.2.4 Laplacian-of-Gaussian filter L6PA

The parameters listed below apply to the Laplacian-of-Gaussian filter.

parameter	value	id
scale	$\sigma^* = \#$	41LN
filter size cutoff	#	WGPM

Table 5.5 — Parameters for Laplacian-of-Gaussian filters. Scale is expressed in physical dimensions, e.g. mm. The filter size cutoff is usually expressed in terms of the scale, e.g. $4\sigma^*$.

5.2.5 Laws kernels JTXT

The parameters listed below apply to the Laws kernels.

parameter	value	id
Laws kernel		41LN
	Level (L3) kernel	B5BZ
	Level (L5) kernel	6HRH
	Edge (E3) kernel	LJ4T
	Edge (E5) kernel	2WPV
	Spots (S3) kernel	MK5Z
	Spots (S5) kernel	RXA1
	Wave (W5) kernel	4ENO
	Ripples (R3) kernel	3A1W
pseudo-rotational invariance	no/yes	O1AQ
response map pooling		SVKW
	average	M1CY
	max	0TAW
	min	DUX4
energy map	no/yes	PQSD
energy map distance	$\delta = \#$	I176

Table 5.6 — Parameters for Laws kernel filters. Unlike the scale parameter of Gaussian and Laplacian-of-Gaussian, the energy map distance δ is defined in voxel units.

5.2.6 Gabor filter Q88H

The parameters listed below apply to Gabor filters.

parameter	value	id
scale	$\sigma^* = \#$	41LN
wavelength	$\lambda^* = \#$	S4N6
ellipticity	$\gamma = \#$	GDR5
filter orientation	$\theta = \#$	FQER
response map		5P3T
	modulus, magnitude or absolute	0J1M
	angle, phase or argument	V530
	real	F7DI
	imaginary	LI3Y
pseudo-rotational invariance	no/yes	O1AQ
response map pooling		SVKW
	average	M1CY
	max	0TAW
	min	DUX4
filter orientation stepping	$\Delta\theta = \#$	XTGK

Table 5.7 — Parameters for Gabor filters. The scale parameter of the Gaussian envelope and the wavelength parameters are expressed in physical dimensions, e.g. mm.

5.2.7 Separable wavelets 25BO

The parameters listed below apply to separable wavelet filters.

parameter	value	id
wavelet family		BPXS
	Haar wavelet	UOUE
	other wavelets	
wavelet filter combination		UK1F
	LL	PPF6
	HL	P60O
	LH	M0S1
	HH	WQ2Z
	LLL	KZAS
	LLH	TY49
	LHL	4KRX
	HLL	UUV9
	LHH	UIT8
	HHL	QTIT
	HLH	8LD2
	HHH	OYUC
wavelet decomposition level	#	GCEK
wavelet decimation		
	decimated transform	PH3R
	undecimated transform	CVCQ
pseudo-rotational invariance	no/yes	O1AQ
response map pooling		SVKW
	average	M1CY
	max	0TAW
	min	DUX4

Table 5.8 — Parameters for separable wavelet filters. Only the Haar wavelet is explicitly described in this reference manual. Other wavelets should be mentioned by name.

5.2.8 Non-separable wavelets LODD

The parameters listed below apply to non-separable wavelet filters.

parameter	value	id
wavelet family		389V
	Shannon wavelet	GWM2
	Simoncelli wavelet	PRT7
	other wavelets	
wavelet decomposition level	#	GCEK

Table 5.9 — Parameters for non-separable wavelet filters. The Shannon and Simoncelli wavelets are explicitly described in this reference manual. Other wavelets should be mentioned by name.

5.2.9 Riesz transformation AYRS

Riesz transformations have the parameters listed below.

parameter	**value**	**id**
Riesz transformation order l	$(\#,\#)$ or $(\#,\#,\#)$	0C48
Riesz filter steering	no/yes	1SD3
Riesz structure window scale	$\sigma^*_{\text{tensor}} = \#$	41LN

Table 5.10 — Parameters for Riesz transformations. The scale parameter of the Gaussian window used to regularise the structure tensor is expressed in physical dimensions, e.g. mm.

Chapter 6

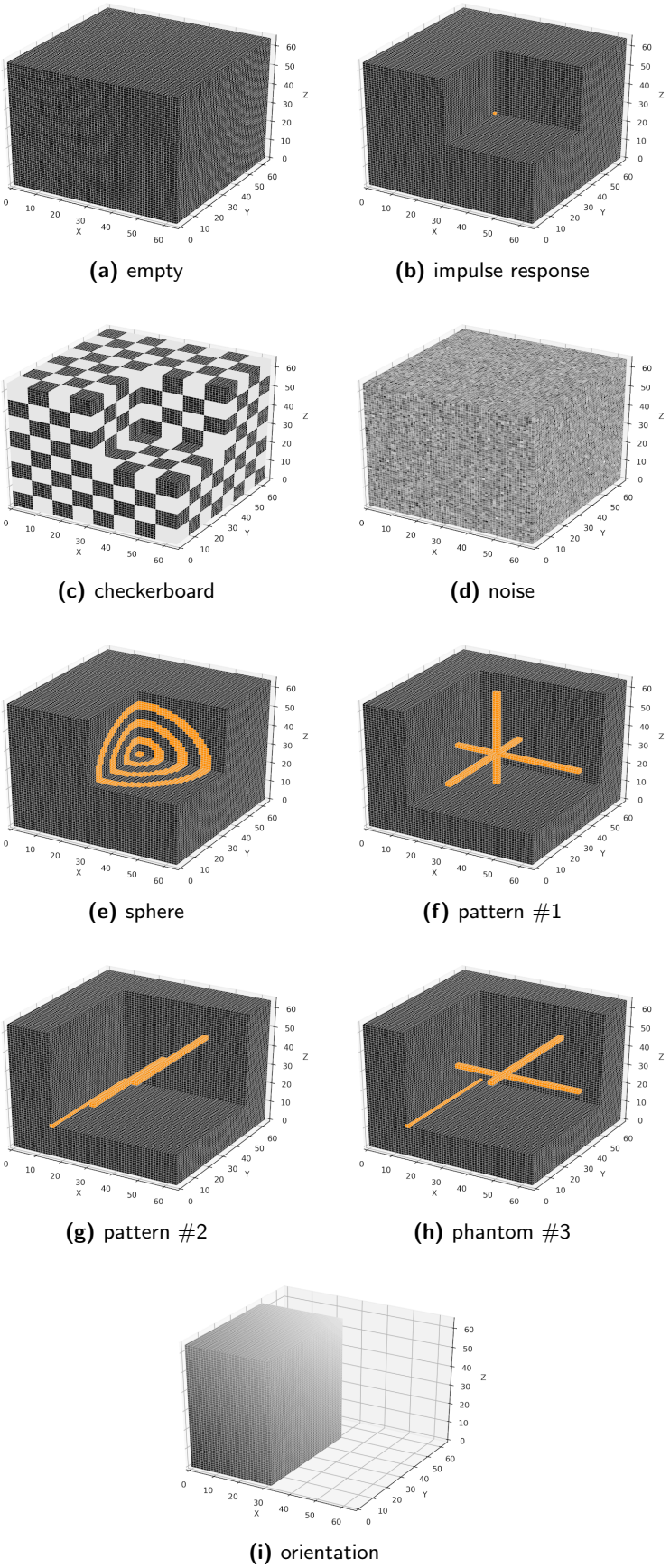
Benchmarking

6.1 Phase 1: Benchmarking filters using digital phantoms

The IBSI developed several $3D$ phantoms to test image filter implementations. All phantoms (but the orientation phantom) have the same dimension and consist of $64 \times 64 \times 64$ isotropic voxels with 2.0 by 2.0 by 2.0 mm spacing. 8-bit voxel intensities in all phantoms fall in the range $[0, 255]$. The phantoms are stored in the NIfTI format. The phantoms are shown in Fig. 6.1. They include:

- **Empty phantom** (`empty.nii.gz`): all voxels have 0 intensity. Intended to investigate the convolution process.
- **Impulse response phantom** (`impulse_response.nii.gz`): all but one voxel have intensity 0. The single remaining centre voxel has an intensity of 255. This allows visualisation of the filter $g[\boldsymbol{k}]$ itself.
- **Checkerboard phantom** (`checkerboard.nii.gz`): alternates between cubic regions with intensity 0 and with intensity 255.
- **Noise phantom** (`noise.nii.gz`): contains Gaussian noise with mean intensity of 127 and a standard deviation of 48. As such it has no inherent structure.
- **Sphere phantom** (`sphere.nii.gz`): consists of four concentric spherical hulls with intensity 255 that are centred on the phantom centre. Thus, the phantom lacks directionality.
- **Pattern #1 phantom** (`pattern_1.nii.gz`): this is the first of three phantoms that involve directionality. Three perpendicular lines (intensity 255) intersect at the centre of the phantom. Along with Pattern #2 and Pattern #3 phantoms, it is intended to investigate filtering methods able to characterise the local organisation of image directions[18].

- **Pattern #2 phantom** (`pattern_2.nii.gz`): this is the second of the directional phantoms. The phantom contains three parallel lines (intensity 255).
- **Pattern #3 phantom** (`pattern_3.nii.gz`): this the last of directional phantoms. The phantom contains three lines (intensity 255), of which two are parallel.
- **Orientation phantom** (`orientation.nii.gz`): Not all filters are rotationally invariant, and therefore the direction along which filters are applied affects the response map. Using the orientation phantom you can check whether the orientation of the image coordinate system in your software matches the orientation expected by the IBSI. The orientation phantom has a dimension of $(32, 48, 64)$ voxels along k_1 (x), k_2 (y) and k_3 (z) axes, respectively. The pixel intensity increases with the distance from the origin, which has an intensity of 0. The most distal voxel has an intensity of 141.



(a) empty **(b)** impulse response

(c) checkerboard **(d)** noise

(e) sphere **(f)** pattern #1

(g) pattern #2 **(h)** phantom #3

(i) orientation

Figure 6.1 — Synthetic phantoms for verifying compliance with reference filter implementation.

The aim of benchmarking the filters using the phantom images is to arrive at a standard implementation. Therefore, filters are applied to the phantoms to create a response maps $h[\boldsymbol{k}]$, which have the same dimensions as the phantoms ($64 \times 64 \times 64$ voxels). Instead of calculating one or more features from each response map (see Section 1.3), the entire response map is to be submitted online to ease debugging and find differences in software implementations. Therefore it is important to note the following:

- The phantom data need to be converted from an integer data type to at least 32 bit floating point precision, prior to filtering.
- Filters need to be applied using the settings provided in Table 6.1. Some settings refer to real-world spacing (in millimetres), instead of voxel spacing.
- The response map needs to be exported in a (compressed) NIfTI format, with at least 32 bit floating point precision. Please make sure that the response map has the correct dimensions.

v6: Added boundary condition for Simoncelli filters (configurations 8.a.1 - 3).

ID	Filter	Phantom	Filter parameters
1.a.1	mean	checkerboard	$3D$ filter, support $M = 15$, zero padding
1.a.2			$3D$ filter, support $M = 15$, nearest value padding
1.a.3			$3D$ filter, support $M = 15$, periodic padding
1.a.4			$3D$ filter, support $M = 15$, mirror padding
1.b.1		impulse	$2D$ filter, support $M = 15$, zero padding
2.a	LoG	impulse	• zero padding • $3D$ filter, scale $\sigma^* = 3.0$ mm, filter size cutoff $4\sigma^*$
2.b		checkerboard	• mirror padding • $3D$ filter, scale $\sigma^* = 5.0$ mm, filter size cutoff $4\sigma^*$
2.c		checkerboard	• mirror padding • $2D$ filter, scale $\sigma^* = 5.0$ mm, filter size cutoff $4\sigma^*$
3.a.1	Laws	impulse	• zero padding • $3D$ filter, E5L5S5 response map
3.a.2			• zero padding • $3D$ filter, E5L5S5 response map • $3D$ rotation invariance, `max` pooling
3.a.3			• zero padding • $3D$ filter, E5L5S5 response map • $3D$ rotation invariance, `max` pooling • energy map, distance $\delta = 7$ voxels
3.b.1		checkerboard	• mirror padding • $3D$ filter, E3W5R5 response map
3.b.2			• mirror padding • $3D$ filter, E3W5R5 response map • $3D$ rotation invariance, `max` pooling
3.b.3			• mirror padding • $3D$ filter, E3W5R5 response map • $3D$ rotation invariance, `max` pooling • energy map, distance $\delta = 7$ voxels

Continued on next page

ID	Filter	Phantom	Filter parameters
3.c.1		checkerboard	• mirror padding • $2D$ filter, L5S5 response map
3.c.2			• mirror padding • $2D$ filter, L5S5 response map • $2D$ rotation invariance, `max` pooling
3.c.3			• mirror padding • $2D$ filter, L5S5 response map • $2D$ rotation invariance, `max` pooling • energy map, distance $\delta = 7$ voxels
4.a.1	Gabor	impulse	• zero padding • $2D$ modulus response map • $\sigma^* = 10.0$ mm, $\lambda^* = 4$ mm, $\gamma = 1/2$ • in-plane orientation $\theta = \pi/3$
4.a.2			• zero padding • $2D$ modulus response map • $\sigma^* = 10.0$ mm, $\lambda^* = 4$ mm, $\gamma = 1/2$ • $2D$ rotation invariance, $\Delta\theta = \pi/4$, `average` pooling • average $2D$ responses over orthogonal planes
4.b.1		sphere	• mirror padding • $2D$ modulus response map • $\sigma^* = 20.0$ mm, $\lambda^* = 8$ mm, $\gamma = 5/2$ • in-plane orientation $\theta = 5\pi/4$
4.b.2			• mirror padding • $2D$ modulus response map • $\sigma^* = 20.0$ mm, $\lambda^* = 8$ mm, $\gamma = 5/2$ • $2D$ rotation invariance, $\Delta\theta = \pi/8$, `average` pooling • average $2D$ responses over orthogonal planes
5.a.1	Daubechies 2	impulse	• zero padding • $3D$ filter, undecimated LHL map – 1^{st} level
5.a.2			• zero padding • $3D$ filter, undecimated LHL map – 1^{st} level • $3D$ rotation invariance, `average` pooling
6.a.1	Coiflet 1	sphere	• periodic padding • $3D$ filter, undecimated HHL map – 1^{st} level
6.a.2			• periodic padding • $3D$ filter, undecimated HHL map – 1^{st} level • $3D$ rotation invariance, `average` pooling
7.a.1	Haar	checkerboard	• mirror padding • $3D$ filter, undecimated LLL map – 2^{nd} level • $3D$ rotation invariance, `average` pooling
7.a.2			• mirror padding • $3D$ filter, undecimated HHH map – 2^{nd} level • $3D$ rotation invariance, `average` pooling
8.a.1	Simoncelli	checkerboard	• periodic padding • $3D$ filter, B map – 1^{st} level
8.a.2			• periodic padding • $3D$ filter, B map – 2^{nd} level

Continued on next page

ID	Filter	Phantom	Filter parameters
8.a.3			• periodic padding • $3D$ filter, B map – 3^{rd} level
9.a	Riesz-transformed LoG	impulse	• zero padding • $3D$ filter, scale $\sigma = 3.0$ mm, filter size cutoff 4σ • $\boldsymbol{l} = (1, 0, 0)$
9.b.1		sphere	• zero padding • $3D$ filter, scale $\sigma = 3.0$ mm, filter size cutoff 4σ • $\boldsymbol{l} = (0, 2, 0)$
9.b.2			• zero padding • scale $\sigma = 3.0$ mm, filter size cutoff 4σ • $3D$ filter, $\boldsymbol{l} = (0, 2, 0)$ • aligned by structure tensor, $\sigma_{\text{tensor}} = 1$mm
10.a	Riesz-transformed Simoncelli	impulse	• zero padding • $3D$ filter, B map – 1^{st} level • $\boldsymbol{l} = (1, 0, 0)$
10.b.1		pattern 1	• nearest value padding • $3D$ filter, B map – 1^{st} level • $\boldsymbol{l} = (0, 2, 0)$
10.b.2			• nearest value padding • $3D$ filter, B map – 1^{st} level • $\boldsymbol{l} = (0, 2, 0)$ • aligned by structure tensor, $\sigma_{\text{tensor}} = 1$mm

Table 6.1 — Filters, parameters and phantoms for comparing and standardising filter implementations. *Note that 2D and 3D rotation invariance for Laws and undecimated wavelet filters (Daubechies 2, Coiflet 1, Haar) is estimated using equivariant right angle rotational representation for separable filters (Appendix A). Pooling refers to pooling over response maps obtained under different orientations. 2D filters are applied in the axial plane by default, except for Gabor filters. For Gabor filters, 2D rotation invariance is estimated by rotating the the filter kernel over multiple orientations in the image plane.*