Rotational Features Extraction for Ridge Detection

Germán González, *Member, IEEE,* François Fleuret, *Member, IEEE,* François Aguet, Fethallah Benmansour, Michael Unser *Fellow, IEEE.*, and P. Fua, *Senior Member, IEEE.*

Abstract

State-of-the-art approaches to detecting ridge-like structures in images rely on filters designed to respond to locally linear intensity features. While these approaches may be optimal for ridges whose appearance is close to being ideal, their performance degrades quickly in the presence of structured noise that corrupts the image signal, potentially to the point where it truly does not conform to the ideal model anymore.

In this paper, we address this issue by introducing a learning framework that relies on rich, local, rotationally invariant image descriptors and demonstrate that we can outperform state-of-the-art ridge detectors in many different kinds of imagery. More specifically, our framework yields superior performance for the detection of blood vessel in retinal scans, dendrites in bright-field and confocal microscopy image-stacks, and streets in satellite imagery.

Index Terms

Rotational features. Ridge Detection. Steerable filters. Data aggregation. Machine Learning.

1 Introduction

Linear structures of significant width, such as roads in aerial images, blood vessels in retinal scans, or dendrites in $3D$-microscopy image stacks, are pervasive. State-of-the-art approaches to detecting them rely on ideal models of their appearance and of the noise. They are usually optimized to find ideal ribbon or tubular structures. However, ridge-like linear structures such as those of Fig. 1 often do not conform to these models, which can drastically impact performance.

In this paper, we use the rotational properties of Gaussian derivatives to achieve rotational invariance as in [2], [18], [23]. However, instead of steering the filters, we rotate the image features to a common reference and replace the optimality criteria by an algorithm that learns from training data. Because this data encompasses deviations from the ideal model, the resulting algorithm is more robust than traditional ones and can be trained to detect not only simple linear structures, but also junctions and crossings. As a result, we obtain better performance than [10], [20] for the detection of blood vessels in retinal scans, neurons in bright-field and confocal microscopy image stacks, and streets in satellite imagery. We chose these two methods as our baseline because they are widely acknowledged as being among the best.

Furthermore, all our computations are based on the output of separable Gaussian filters, which can be implemented very efficiently. This is important for $2D$ and even more so for $3D$ image volumes due to the potentially very large size of the datasets involved. The corresponding code is publicly available [15] and is compatible with the ITK open-source, object-oriented software system for image processing, segmentation, and registration [17].

This paper expands on our earlier ones [14], [16] on the same topic. We first discuss related work on linear structure detection. Then, we briefly review the concept of steerable filters, which underpins our rotational features, and introduce our methodology for both $2D$ and $3D$ image data. Finally, we present comparative results obtained on different datasets.

- *This project was supported in part by the ERC grant MicroNano.*
- *G. González F. Benmansour and P. Fua are with the Laboratory of Computer Vision at École Polytechnique Fédérale de Lausanne, Switzerland.*
  F. Fleuret is at the Idiap Research Insititute in Martigny, Switzerland.
  F. Aguet is at the Department of Cell Biology, Harvard Medical School, Boston.
  M. Unser is the head of the Biomedical Imaging Group at École Polytechnique Fédérale de Lausanne, Switzerland.
- *E-mail: pascal.fua@epfl.ch.*

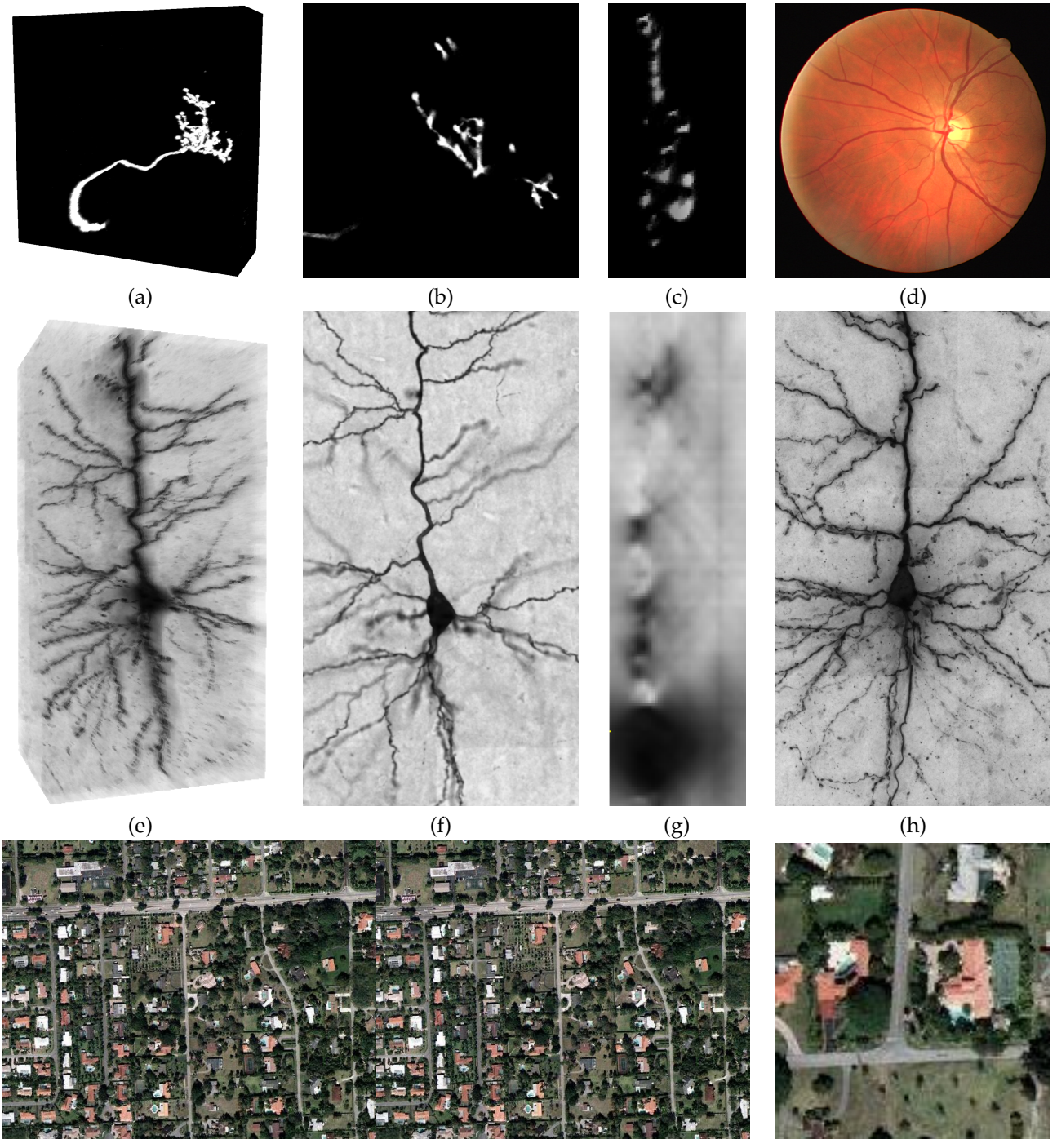


Fig. 1: Ridge-like linear structures come in many flavors. This figure depicts the datasets we have experimented with, sorted in order of increasing difficulty. (a,b,c) Confocal microscopy image-stack showing axons of the olfactory bulb of the drosophila fly labeled with green fluorescence protein [5]. The first image is a maximum intensity projection while the other two are XY and XZ slices respectively. Note the clutter in some portions of the image. (d) Retinal scan [29]. (e,f,g,h) Neuron imaged using bright-field microscopy. The first image is a minimum intensity projection of the image-stack. The next two are XY and XZ slices respectively. The fourth one is Z-projection. Note the cone-shaped blur in the XZ projection. (i,j) Aerial Image of a suburban neighborhood and zoomed-in window showing the small contrast between roads and grassy areas. For the purpose of linear structure detection, the house and trees can be considered as structured noise.

2 Related Work

Current approaches to finding linear structures that are not simple edges, such as those of Fig. 1, fall into three main categories. Some rely on models of an ideal ridge-like structure to derive optimal filters, while others involve computing the Hessian matrix centered on individual pixels and depending on its eigenvalues to classify the pixel on the basis of how cylinder-like the local intensities are. Probabilistic approaches to detect non-ideal linear structures exist, but they usually rely on low-dimensional image descriptors, therefore constraining their performance. We briefly discuss these techniques below. For a more extensive review of vessel extraction techniques, we refer the interested reader to [19], [21].

2.1 Hessian-Based Approaches

Hessian-based approaches to filament detection model linear structures as elongated ellipses or ellipsoids. This involves computing the eigen-decomposition of the Hessian matrix at individual pixels and using its eigenvalues to classify pixels as filament-like or not [10], [26], [30]. The Hessian matrix for a given pixel is constructed by convolving the local image patch with the set of second order Gaussian derivative filters. To find filaments of various widths, a range of variances for the Gaussian is used and the most discriminant one is selected. This results in a smoothly decreasing response in the perpendicular direction of the ridge, since wide detectors respond to thin ridges in an unlocalized manner.

2.2 Optimal Filtering

Optimal filters attempt to find linear image structures by following the criteria for optimal edge detection outlined in [7]. Methods following this approach include the Canny detector itself [7] and optimal convolution filters [24].

An elegant and computationally efficient approach to detecting edges and ridges at arbitrary orientations is to use steerable filters [11]. The underlying principle is that the response of the filter at any orientation can be calculated as a linear combination of those of a set of basis filters, thus avoiding the expensive convolutions with orientation-specific 2D kernels. A special case of steerable filters is the linear combination of Gaussian derivatives up to a given order [2], [18], [23].

Along similar lines, the Oriented Flux Filter [20], obtained by convolving the second derivatives of the image with the indicator of a sphere, is a steerable filter designed for detecting ideal sharp ridges. Compared to Hessian-based detectors, the Oriented Flux Filter is simpler to normalize over scales and is less sensitive to the presence of adjacent features.

However, the criteria used to derive the steerable filters for ridge detection assumes ideal models of the ridges and noise. In theory, more realistic models could be used without changing the overall approach, but no generic strategy has been offered as to how this could be done.

2.3 Detection Techniques for Ridge Tracking

Ridge tracking and active testing techniques follow the ridges from seed points that can be provided manually or automatically detected [3], [12]. Such techniques can use complex models to detect ridges, since the function is not evaluated densely over the whole image. The fact that intensity changes inside and outside the filaments has been explicitly exploited in such a context by explicitely testing for them [12], locally convolving the image with differential kernels [4], finding parallel edges [3] and fitting generalized cylinders [27] or *superellipsoids* [32] to the vessel based on its contour integral. All these methods, however, assume image regularities that are present in well-behaved images but not necessarily in noisier ones. Furthermore, they often require careful parameter tuning, which may change from one dataset to the next. Due to the cost of optimizing the meta-parameters of the methods to each pixel under consideration, these techniques are not suitable for vessel detection and segmentation.

2.4 Probabilistic Approaches

Probabilistic approaches able to learn whether a pixel belongs to a filament or not have been recently applied to the problem. Instead of assuming the filaments to have a regular shape, they aim at learning their appearance from the data. In [1], the eigenvalues of the structure tensor are represented by a mixture model whose parameters are estimated via Expectation Maximization. Support Vector Machines operating on the Hessian's eigenvalues have also been used to discriminate between filament and non-filament pixels [25]. Similarly, Probabilistic Boosting Trees on rotational features have been demonstrated for vessel segmentation [28].

The two latter approaches, [25], [28] are the closest to ours because they also rely on a learning paradigm. However, the generalization ability of [25] is limited by the fact that it still relies on the eigenvalues of the Hessian, a low-dimensional descriptor. By contrast, we train our classifier directly on the space of higher order Gaussian derivatives, thereby allowing it to handle structures whose shape is more variable. The main difference between our rotational features and those of [28] is that ours are estimated densely around the pixel under analysis, while theirs only are evaluated at sparsely sampled points and used to verify hypotheses made by another algorithm.

3 METHODOLOGY

Our goal is to devise an algorithm that detects filament-like structures of interest at any orientation or scale while rejecting the noise present in the images. Our algorithm goes through the following steps. For each pixel we compute multi-scale rotational features using Gaussian derivatives of various widths. Then, we train an SVM classifier on those feature vectors rotated to a reference orientation. This classifier is used to classify filament-like structures at any orientation. With no loss of generality, we will derive the formulae in the 3D case, letting the 2D case be a subset of it.

3.1 Steerable Filters

Steerable filters were introduced as an efficient means to compute filters that can be rotated to any orientation for a small computational cost [11]. We describe them briefly below and refer the interested reader to [18] for more details.

A steerable filter based detection of a feature g in an image I at a given orientation $\mathbf{\Theta}$ and position $\mathbf{u}$, is formulated as

$$r(\mathbf{u},\mathbf{\Theta}) = (I(\mathbf{u}) * g(R^{\mathbf{\Theta}}.))(\mathbf{u}), \quad g(R^{\mathbf{\Theta}}\mathbf{u}) = \sum_{k=1}^{K} \beta_k(\mathbf{\Theta}) f_k(\mathbf{u}) \;, \tag{1}$$

$$r(\mathbf{u},\mathbf{\Theta}) = \langle \beta(\mathbf{\Theta}), (I * \mathbf{f})(\mathbf{u}) \rangle \;, \tag{2}$$

where $\mathbf{\Theta}$ parametrizes the orientation of the feature template g, $R^{\mathbf{\Theta}}$ is a rotation matrix, r is the response and $\langle .,. \rangle$ denotes the dot product between vectors. The functions $\beta_k(\mathbf{\Theta})$ are trigonometric polynomials that interpolate the templates $f_k(\mathbf{u})$, and $\mathbf{f}(\mathbf{u}) = (f_1(\mathbf{u}), \ldots, f_K(\mathbf{u}))$. This decomposition decouples the rotation of the filters from the convolution and lets us write the filtering at any orientation as the dot product of the rotated coefficients $\beta(\mathbf{\Theta})$ with the convolution of the image with the templates $\mathbf{f}$, as in Eq. 2. This makes estimating the responses at arbitrary orientations computationally efficient.

3.2 Steerablility of Gaussian Derivatives

The best known class of steerable filters, and the ones used in this paper, are Gaussian derivatives and their linear combinations [18]. We constrain the Gaussian function to have a diagonal covariance matrix so that the Gaussian function and its derivatives are separable in all dimensions. This limits the computational cost of the convolutions to Nd instead of N^d, where N is the size of the kernel and d the image dimensionality.

As shown in [11], any polynomial multiplied by a radially symmetric function is steerable. Any derivative of a radially symmetric Gaussian function is a polynomial that multiplies the original Gaussian function:

$$\frac{\partial^m}{\partial x^m}\frac{\partial^n}{\partial y^n}\frac{\partial^{q-m-n}}{\partial z^{q-m-n}} G^\sigma(\mathbf{u}) = p^m(x)p^n(y)p^{q-m-n}(z)G^\sigma(\mathbf{u}) \;, \tag{3}$$

where $p^i(u)$ is the Hermite polynomial of order i over the variable u. Since polynomials are steerable and $G^\sigma(\mathbf{u})$ is radially symmetric, the Gaussian derivatives are steerable.

3.3 Rotational Features

The feature vector we use in the algorithm is the value at $\mathbf{u}$ of the convolution of the image by the Gaussian derivatives up to order M at a given scale σ

$$\mathbf{v}^\sigma(I,\mathbf{u}) = \left(I * \left[G^\sigma_{0,0,1}, G^\sigma_{0,1,1}, G^\sigma_{1,0,1}, G^\sigma_{0,0,2} \cdots G^\sigma_{M,0,M}\right]\right)(\mathbf{u}) \;, \tag{4}$$

where $G^\sigma_{m,n,p}$ denotes the Gaussian derivative m times over x, n times over y and $p-m-n$ times over z. This vector has a dimension of $K = (M+1)(M+2)/2$. To achieve scale independence, we extend the feature

vector of each point by adding features to it at s different scales. The full feature vector can then be written as $\mathbf{v}(I,\mathbf{u}) = [\mathbf{v}^{\sigma_1}(I,\mathbf{u}), \cdots, \mathbf{v}^{\sigma_s}(I,\mathbf{u})]$ and its dimension is $D = sK$.

Following [18], any vector in the space defined by the Gaussian derivatives can be rotated or steered. By rotation, we mean that the shape of the function at any orientation can be defined as a linear combination of the Gaussian derivatives. The coefficients of that linear combination depend on the desired orientation. More precisely, if for an image I, angle $\mathbf{\Theta}$, and location $\mathbf{u}$ in the image frame, $I^{\mathbf{\Theta}}$ denotes the image after rotation and $\mathbf{u}^{\mathbf{\Theta}}$ the location in the rotated image frame, we have

$$\forall\sigma, \forall\mathbf{\Theta}, \quad \exists\mathcal{R}^{\mathbf{\Theta},\sigma} \in \mathbb{R}^{d\times d}, \quad \text{such that,} \quad \forall I, \forall\mathbf{u}, \quad \mathbf{v}^{\sigma}(I^{\mathbf{\Theta}}, \mathbf{u}^{\mathbf{\Theta}}) = \mathcal{R}^{\mathbf{\Theta},\sigma}\mathbf{v}^{\sigma}(I,\mathbf{u}), \tag{5}$$

where $\mathcal{R}^{\mathbf{\Theta},\sigma}$ is a suitable steering matrix. Hence, extracting the feature vector at any random orientation does not require the evaluation of new linear filter responses, but simply multiplying the vector $\mathbf{v}^{\sigma}(I,\mathbf{u})$ by a matrix parametrized by the $\mathcal{R}^{\mathbf{\Theta},\sigma}$ matrix, which is parameterized by both $\mathbf{\Theta}$ and σ. Its coefficients are derived in the Appendix. Since this rotation of the feature vector is performed at all scales, let $\mathcal{R}^{\mathbf{\Theta}}$ denote the block matrix that rotates the multi-scale feature vector.

3.4 Learning the Shape of the Filaments

Since we can rotate all feature vectors to a canonical orientation, we can train a single generic classifier as follows. We randomly select N triplets (image, location, orientation) corresponding to filament structures

$$\{(I_1,\mathbf{u}_1,\mathbf{\Theta}_1),\ldots,(I_N,\mathbf{u}_N,\mathbf{\Theta}_N)\}, \tag{6}$$

and N triplets corresponding to non-filaments

$$\{(I_{N+1},\mathbf{u}_{N+1},\mathbf{\Theta}_{N+1}),\ldots,(I_{2N},\mathbf{u}_{2N},\mathbf{\Theta}_{2N})\}. \tag{7}$$

Then, from the D-dimension feature vectors extracted at these points

$$\forall n,\ \mathbf{v}_n = (\mathcal{R}^{\mathbf{\Theta}_n})^{-1}\,\mathbf{v}(I_n,\mathbf{u}_n) \tag{8}$$

we define a training set, the first N samples of which are of class 1 and the last N of class 0

$$\{(\mathbf{v}_1,1),\ldots,(\mathbf{v}_N,1),(\mathbf{v}_{N+1},0),\ldots(\mathbf{v}_{2N},0)\}\ . \tag{9}$$

From that labeled sample set, we train an Support Vector Machine (SVM) [8] of the form

$$h:\mathbb{R}^D\rightarrow\mathbb{R};\quad (\mathbf{v}) = \sum_{n=0}^{N} a_n\,\kappa(\mathbf{v}_n,\mathbf{v}) + b\ , \tag{10}$$

where κ is the standard Gaussian kernel with variance ν, which is obtained using n-fold cross validation while training.

This strategy of training a classifier common to all orientations is a direct application of the idea of data aggregation through stationary features [9]. We parametrize the features with a complex pose instead of training several classifiers dedicated to constrained subsets of samples. Doing so, we avoid both the computational overhead of training several classifiers and the over-fitting due to the fragmentation of the sample set.

3.5 Detecting Filaments

The image features we use are the same as those proposed in [18] but, because the SVM is nonlinear, there is no analytical criterion to decide at which orientation we will get the maximum response of the classifier of Eq. 10. For an image I and location $\mathbf{u}$, the orientation $\mathbf{\Theta}$ at which the filament appears is a hidden variable that needs to be estimated. As a result, we have to sample the space of possible orientations and find the one that produces the greatest SVM response

$$\psi(I,\mathbf{u}) = \max_{\mathbf{\Theta}} h\left((\mathcal{R}^{\mathbf{\Theta}})^{-1}\,v(I,\mathbf{u})\right)\ . \tag{11}$$

For 2D images, we explicitly perform this maximization over the orientation space as in earlier work [16]. However, when dealing with 3D image stacks, sampling the whole 3D orientation space becomes prohibitively expensive. Instead, we estimate the orientation at each pixel using a standard method. We then evaluate the SVM response for that orientation only

$$\hat{\psi}(I, \mathbf{u}) = h\left((\mathcal{R}^{\hat{\mathbf{\Theta}}})^{-1}\, v(I, \mathbf{u})\right), \tag{12}$$

where $\hat{\mathbf{\Theta}}$ is taken to be the direction of the eigen-vectors of the modified Hessian matrix [2] at the location $\mathbf{u}$. We found empirically that orientations computed in this way yield better results than those obtained using other methods.

4 RESULTS

In this section we compare the performance of our approach to detection of ridge-like linear structures against that of [10] and [20], two of the best known algorithms in the field. To this end, we use the five kinds of images depicted by Fig. 1. They are $2D$ images of retinal scans and roads in satellite imagery, $2D$ images and $3D$ image stacks of dendritic networks in bright-field microscopy, and $3D$ image-stacks of axons in confocal microscopy. Fig. 2 and Table 1 summarize our quantitative evaluation and support the two following conclusions:

- For the purpose of extracting the centerlines of the linear structures, whether thick or thin, our algorithm outperforms the other two.
- For the purpose of finding the width of the linear structures, the other two algorithms do slightly better than ours on clean data but noticeably worse on noisier data.

This makes sense since both the methods of [10] and [20] are optimized to find ideal structures whereas our algorithm learns the irregularities of the data. When operating at low recall rates, when only thick filaments are detected, this entails a small penalty because our classifier has expanded some of its descriptive power to be as good at finding the thin than the thick ones.

In the remainder of this section we first describe our approach to training our classifier and discuss our evaluation methodology. We then analyze our results on the five images types of Fig. 1.

4.1 Training the Classifier

Each dataset includes of at least two fully annotated images. The ground truth data was produced by experts in one of two ways:

1) They traced the relevant structures as generalized cylinders by specifying centerlines and corresponding widths or radii. Note that even experienced operators often make mistakes estimating these and that the values they provide cannot be considered as perfectly reliable.
2) They provided segmentation masks corresponding to pixels deemed to be within the ridges. We then skeletonized them to obtain the centerlines.

From the ground truth data of the different datasets we harvested positive and negative samples as follows: positive samples were collected from the centerline of the ground truth and their orientation estimated from it. Negative samples were chosen randomly from the outside of the ridges, half of them close to the filaments and the rest from random locations elsewhere in the image. We assigned to negative samples the orientation corresponding to the principal eigenvalue of the Hessian matrix at that location. For each sample ground truth location $\mathbf{u}_n$ we computed a feature vector $\mathbf{v}_n$ and rotated it inversely to its corresponding orientation $\mathbf{\Theta}_\mathbf{n}$. The annotated data was divided into disjoint training, and test sets, leaving at least one whole image for testing.

The training set for each image type contains 5000 positive and 5000 negative samples. It was used to train the SVM and the meta-parameters, namely the kernel variance ν and the regularization parameter C were found by doing n-fold cross validation. We use derivatives up to order $M = 4$, which we had shown to yield better results than stopping at order $M = 2$ in earlier work [14].

4.2 Evaluation Methodology

Ridge detection usually is the first step in either a segmentation or a reconstruction pipeline, such as those of [31], [33]. Depending on the final application there are different requirements on the ridge detection algorithm. For instance, if the goal is to measure the area of blood vessels in retinal scans, the algorithm should produce a perfect image segmentation. By contrast, if it is to recover the network topology, the algorithm should enhance both thin and thick vessels alike, since missing a thin vessel is worse than incorrectly estimating the width of a thick one.

To asses the performance of the algorithms in both scenarios, we analyzed them using two different metrics that we describe below. The first one is designed to measure performance when centerlines are what matters and the second when correct segmentations are required.

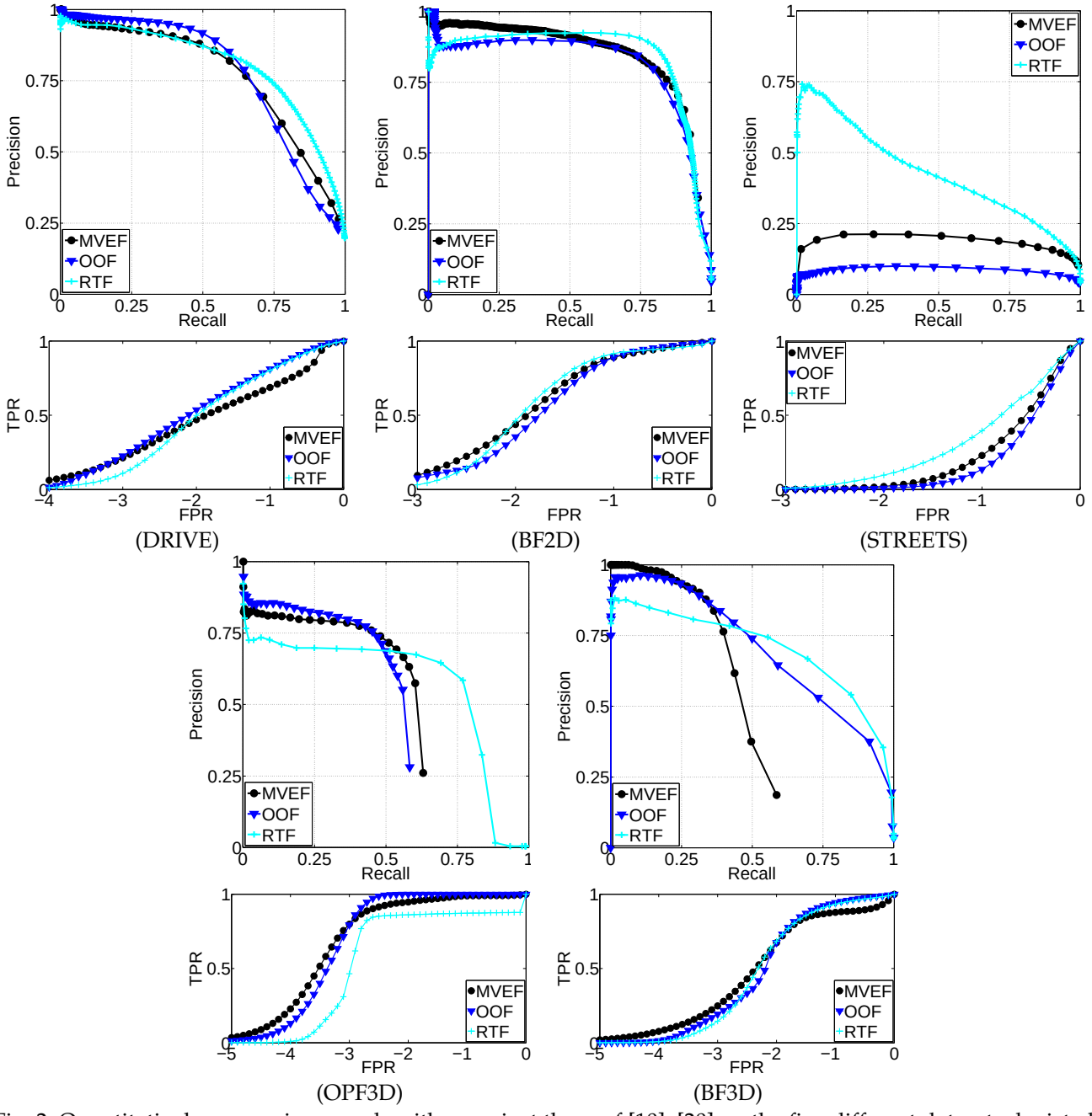


Fig. 2: Quantitatively comparing our algorithm against those of [10], [20] on the five different datasets depicted by Fig. 1. **Top row:** $2D$ datasets. **Bottom row:** $3D$ datasets. For each dataset, we plot at the top the precision-recall curves of Section 4.2.1 and below the ROC curves of Section 4.2.2. The cyan curves are those produced by our Rotational Features, the dark blue ones by the Hessian-based algorithm [10], and the black ones by the Oriented Flux Filter [20]. We denote them as RTF, MVEF, and OOF in the following figures.

4.2.1 Centerline Detection Metric

To evaluate the fraction of ridges that our method correctly detects, we use the metric of [22], which was initially proposed to assess boundary detection techniques. It is also appropriate for ridge detection because ridges are one-dimensional elongated objects embedded in a higher-dimensional image.

The original article [22] introduced a well-defined criterion to compare two different delineations, which we adapt for our purposes as follows. We first perform non-maximum suppression on the image produced by the detector. After thresholding, we match the resulting image against the ground-truth centerline using a bipartite graph and a perfect graph matching algorithm. Given that match, we can identify true positives as matches in the bipartite graph between ridge pixels in the test image against ridge pixels in the ground

truth. False positives are matches of the bipartite graph against extra nodes in the ground truth and false negatives are matches of the ground truth against extra nodes in the test image. This matching is repeated for different thresholds to plot precision-recall curves such as those at the top of each column in Fig. 2. Note that the non-maxima suppression step ensures that this evaluation method gives mistakes on thin or thick filaments equal importance.

To condense each one of these curves to a single number, we proceed as in [22] and simply compute the maximum *F-Measure* for the different detectors. The *F-Measure* is the harmonic mean of the precision and recall for a given point of the detector performance. That is

$$F = 2 \times \frac{p \times r}{p + r} \, , \tag{13}$$

where p stands for the precision of the classifier at a given recall r. The maximum *F-Measure* for the different classifiers is shown in Table 1. This provides an overall estimate of the algorithm's performance. Since a perfect algorithm would consistently yield recall and precision values of 1.0, the best F-Measure that can be obtained also is 1.0. For any other algorithm, the closer the F-Measure is to 1.0, the better.

All these precision-recall values depend on one parameter, the maximum distance allowed between one pixel in the ground truth and a pixel in the test image to be considered a match candidate. In the plots of Fig. 2, this parameter is taken to be 3 pixels and we checked that increasing its value to 7 pixels does not change the ordering of the curves.

Dataset	MVEF	OOF	RTF
Brightfield 3D	0.5234	0.6169	**0.6814**
OPF 3D	0.5780	0.6090	**0.6683**
DRIVE	0.7049	0.7101	**0.7449**
Brightfield 2D	0.8010	0.7970	**0.8389**
Road	0.2997	0.1627	**0.4591**

TABLE 1: F-Measure obtained by computing the area under the three precision-recall curves that appear at the top for each one of the five datasets in Fig. 2. A perfect score would be 1.0. Overall, our method outperforms [10], [20] in all five cases.

4.2.2 Segmentation Metric

As in [29], we use this method to compare the segmentation produced by our algorithms against the ground truth segmentation. When the ground truth consists of a set of traces with widths, we render such traces on the image to obtain a segmentation mask. The Receiver Operating Characteristic curve (ROC) is computed by thresholding the detection and finding true positives (points over the threshold that belong to the segmentation mask), false positives (points over the threshold that do not belong to the segmentation mask), true negatives (points under the threshold that do not belong to the mask), and false negatives (points under the threshold that belong to the mask). The threshold is increased from the minimum value in the detection image to the maximum value.

At the bottom of each column in Fig. 2, we plot the true positive rate as a function of the false positive rate for each one of the three algorithms we tested.

4.3 Performance Comparison

We now turn to comparing our algorithm's performance, denoted by the RTF acronym in the figures, against a Hessian-based one designed for Multiscale Vessel Enhancement Filtering [10] and the Oriented Flux Filter [20], that we denote as MVEF and OOF respectively, on each one of our five datasets.

4.3.1 Blood Vessels (DRIVE)

The DRIVE dataset [29] consists of 40 retinal scans. In each one, the background is mostly uniform and the vessels appear as dark structures. However, a clear circular structure, close to the root of the tree is a localized source of noise, as can be seen in Fig. 1(d). The edges of this structure cause problems to all three algorithms we have tested. The scans are split into two sets, one for training and the other for testing. The training set was segmented by one expert, while the test set was segmented by two experts to allow for inter-expert variability analysis.

The resulting precision-recall and ROC curves are shown in the first column of Fig. 2. The corresponding F-Measures on the third line of Table 1 indicate that our method outperforms the other two in terms of the centerline detection metric. More specifically, the precision-recall curves of Fig. 2(DRIVE) show our method

performing best for recall values greater than 0.6, which are the relevant ones in practice. For smaller recall values, our method performs as well as MVEF [10] but worse than OOF [20]. In terms of the segmentation metric, our algorithm performs better than MVEF and similarly to OOF for high true positive rates. This dataset is the cleanest $2D$ dataset in which we evaluated our algorithm, and the ridges often conform to the ideal models for which MVEF and OOF were designed for. When the false positive rates fall below 1%, both competing methods outperform ours. However, at that rate, less than 40% of true positives are detected, which renders the results almost useless for all three methods.

Figure 3 shows the filter responses and segmentation results on a small window of a test image. The segmentations are obtained by thresholding the detections to obtain a false positive rate of 3%. OOF and our RTF algorithm produce virtually indistinguishable results, while MVEF clearly yields fewer true positives. An interesting artifact of MVEF is the high response on the sides of the thick vessels. This is caused by the non-convex M-shape profile of such vessels. MVEF detects two small ridges instead of a bigger one. OOF suffers from the same problem but to a lesser extent, while RTF responds much more homogeneously.

The fact that both competing methods outperform ours at low false positive and low recall rates derives from our training scheme being biased towards thin filaments. Since the ground truth data contains more thin vessels than thick ones and our selection mechanism for positive training examples does not take this into account, more thin than thick vessels end up being considered. As a result, the classifier learns to distinguish them better. This shows up at low recall rates, which is precisely when only the thick filaments are detected. In theory this could be fixed using a more sophisticated sampling scheme but, since the problem shows up only outside of the reasonable operating range, we do not consider this as crucial.

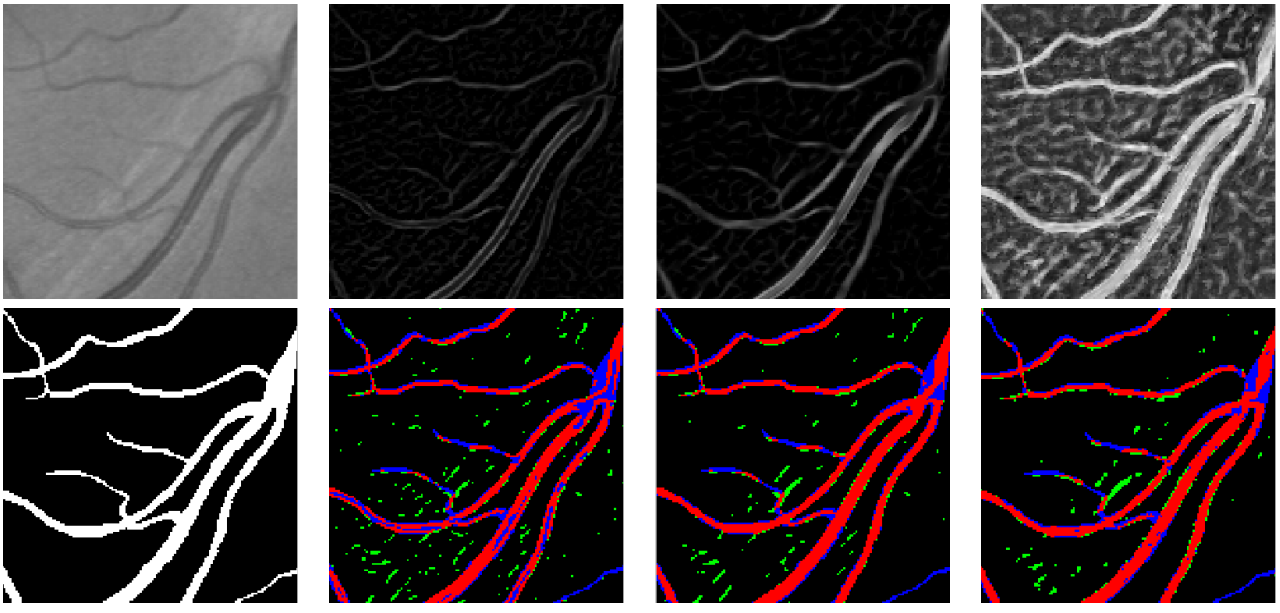


Fig. 3: Close-up of a small-window of an image of the DRIVE dataset. **Top row:** original image and output of the filters. **Bottom row:** Ground truth and segmentations obtained for a False Positive Rate of 3%. True positives are displayed in red, false positives in green and false negatives in blue. In this dataset the segmentation produced by OOF and RTF are virtually indistinguishable and better than the one produced by MVEF. This is consistent with the ROC curves at the bottom of Fig. 2(DRIVE).

4.3.2 Neurons in Brightfield Microscopy (BF3D and BF2D)

The neuron images of Fig. 1(e-h) were obtained from rat brains at EPFL's Neural Microcircuitry Laboratory. The neuron was dyed with byocitin and then imaged with a bright-field microscope at different tissue depths, creating a $3D$ image stack. Several artifacts appear in these images due to the irregularities of the staining process, and the large non-Gaussian blur produced by the image acquisition technique. As a result, many interesting filaments appear as faint structures, filaments with abrupt changes in width are severely blurred, and the out-of-plane information heavily corrupts the image, as shown in Fig. 4 in an XY cut of the $3D$ image and on Fig. 1(g) in an XZ plane. In this paper, we use the original $3D$ data to evaluate the performance of the $3D$ rotational features and also use the $2D$ minimum intensity projection of the $3D$ image to evaluate their $2D$ counterparts.

3D Image Stacks The artifacts discussed above make MVEF [10] and OOF [20] fire at out-of-focus locations where there is no dendrite in the ground truth data. Our algorithm learns to reject such non-homogeneous blur and thus produces a response that is more constrained to the plane in focus. This effect can be seen in the comparative detections of Fig. 4. Our algorithm also learns that structures of irregular width can still be dendrites and, thus, recovers them better than MVEF or OOF. As was the case for blood vessels and can be seen from the curves at the top of Fig. 2(BF3D), our algorithm outperforms them in terms of centerline detection for true positive rates greater than 0.6, resulting in the improved F-measure of the second line of Table 1. The ROC curves at the bottom of Fig. 2(BF3D) also indicate better performance than MVEF and similar performance to OOF at high true positive rates.

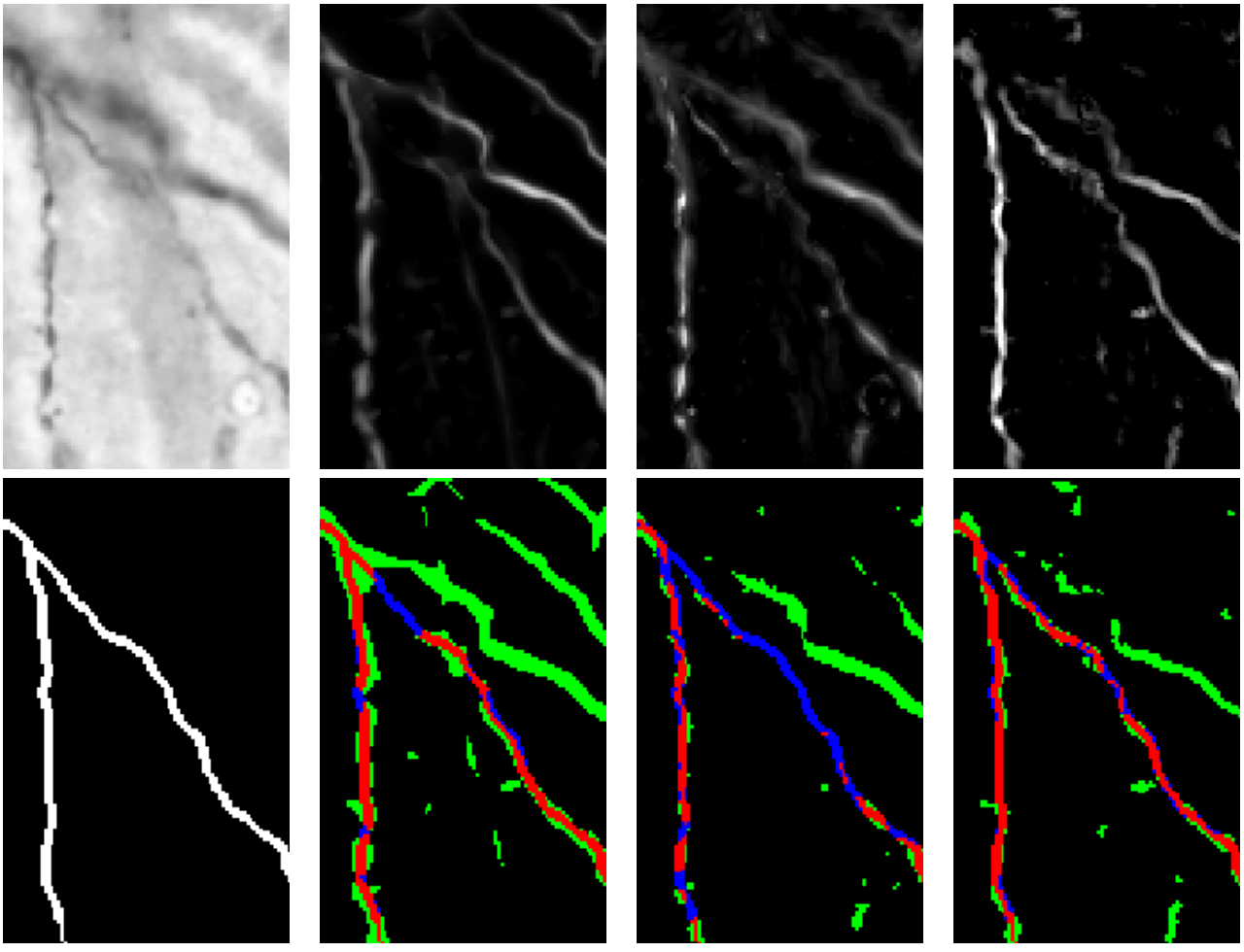


Fig. 4: Close-up on a detail of the dendritic tree from a slice in the bright-field microscopy dataset. **Top row:** Original image and filter responses. **Bottom row:** ground truth and segmentation for an FPR of 1%. As in Fig. 3, we display true positives in red, false positives in green and false negatives in blue. Note the lower response of RTF to out-of-focus dendrites. Also, thanks to its training, RTF can adapt and detect the irregular shape of the vertical ridge better than the competing algorithms.

2D Minimum Intensity Projections As can be seen in Fig. 5, these projections exhibit the same global appearance as individual slices, but with the added complexity that filaments that were separated in $3D$ now appear to be crossing each others. Nevertheless, the rotational features are versatile enough for our classifier to learn the appearance of such crossings, something the closed-form MVEF and OOF detectors cannot do. Therefore, our method not only outperforms them in terms of the centerline detection metric, as can be seen from the curves at the top of Fig. 2(BF2D) and corresponding F-measures of Table 1, but also in terms of the ROC curves at the bottom of Fig. 2(BF2D).

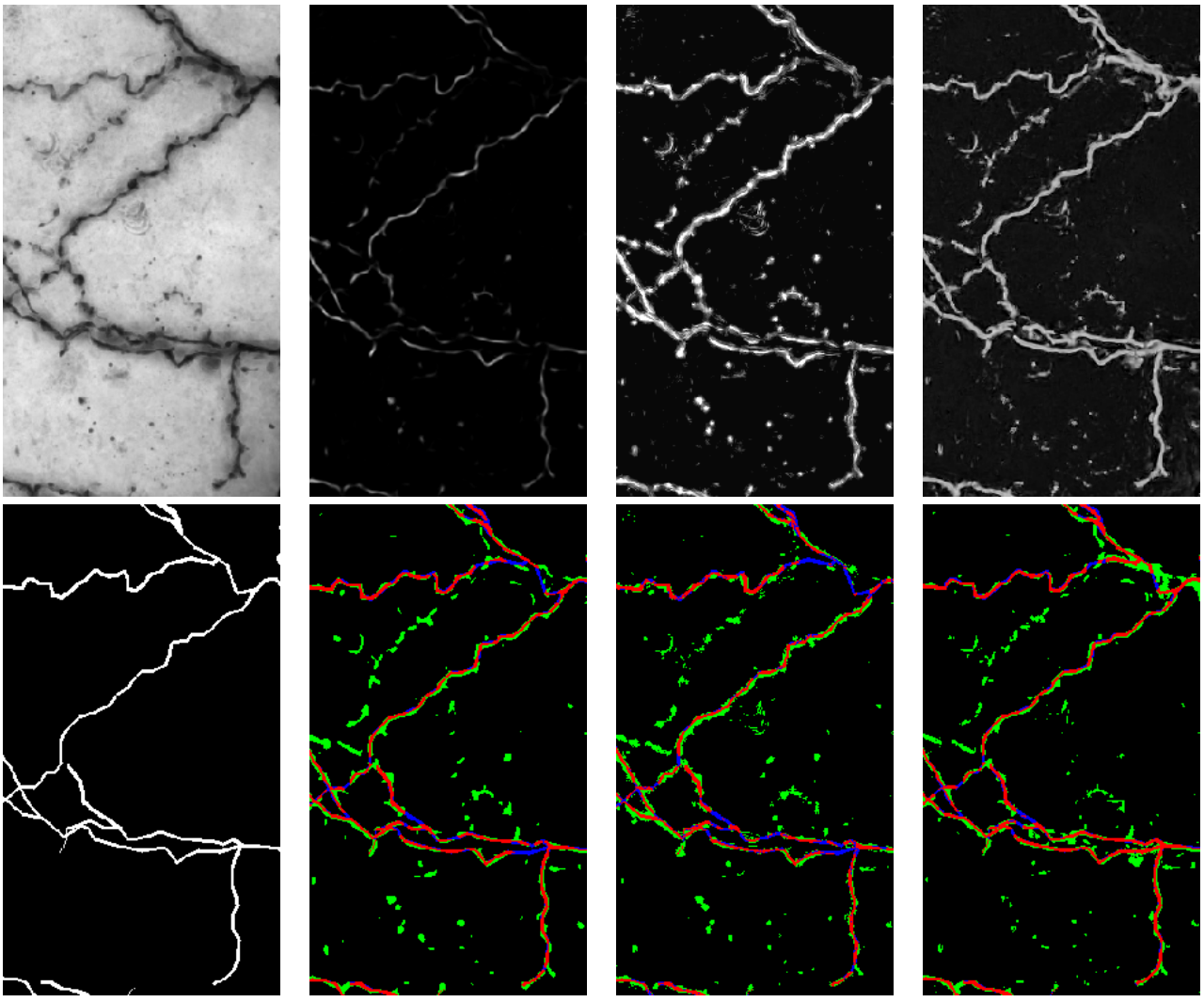


Fig. 5: Close-up on a window from the $2D$ version of the bright-field dataset. **Top row:** Original image and filter responses. **Bottom row:** ground truth and segmentation for an FPR of 3%. Please note that RTF produces more true positives than the others, especially on challenging parts of the image such as the parallel ridges or the bifurcations and crossings.

4.3.3 Confocal Microscopy (OPF3D)

This is a dataset that was provided by the DIADEM challenge [5]. It consists of axons of the olfactory bulb of the drosophila fly labeled with green fluorescence protein (GFP). The database consists of six $3D$ image stacks, each of them containing one axon. The images were obtained with a two channel confocal microscope, resulting in low noise levels with little blur, as shown in Fig. 1 (a-c). The main challenge of this dataset is that some regions can be cluttered. The neurons are traced using Neurolucida [6] and Amira [13], which often results in an over-estimation of axonal width, as shown in Fig. 6. Since our detector responds negatively to the edges of the filaments and the ground truth data over-estimates their width, our performance in the segmentation ROC curve is worse than that of the other two methods. However, as can be seen at the top of Fig. 2(OPF3D), when we perform the centerline-based evaluation, our performance is better at high false positive rates.

4.3.4 Streets in Aerial Images (STREETS)

Detecting streets in aerial images was the most challenging task for all three algorithms we tested. We detected filaments in the street images in the same manner as the vessel and neuron images. As one might expect, the results are worse in this dataset, as shown in Fig. 2(STREETS). Nevertheless, our algorithm outperforms MVEF and OOF even more convincingly than before. This is because it is much less sensitive to the structured

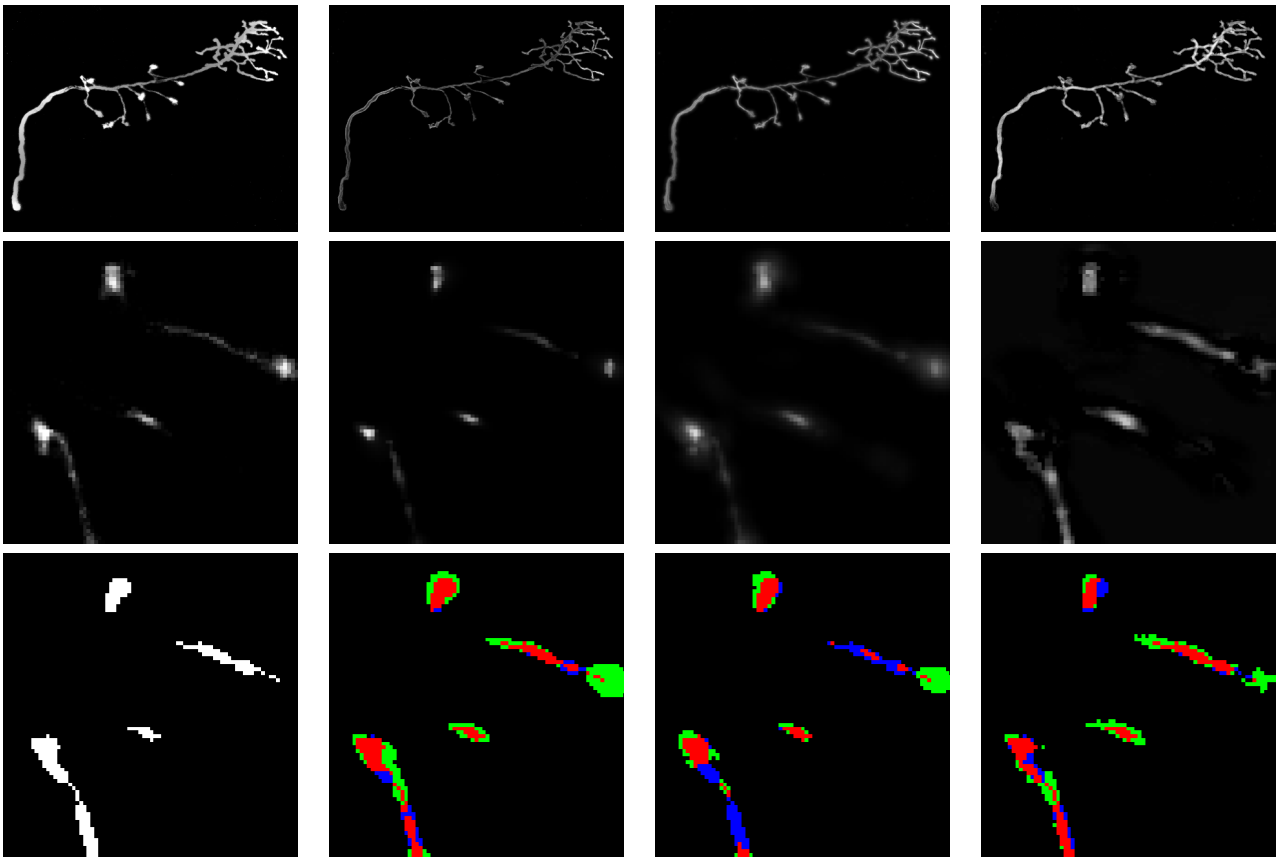


Fig. 6: Olfactory Projection Fibers dataset. **Top row:** full image. **Middle row:** detection in an XY layer. **Bottom row:** segmentation for a false positive rate of 1%. The three algorithms perform really well on this clean image. Note that the RTF response is tighter to the axon and the ground truth sometimes over-segments the image. This causes our score to decrease in the ROC curve of Fig. 2(d).

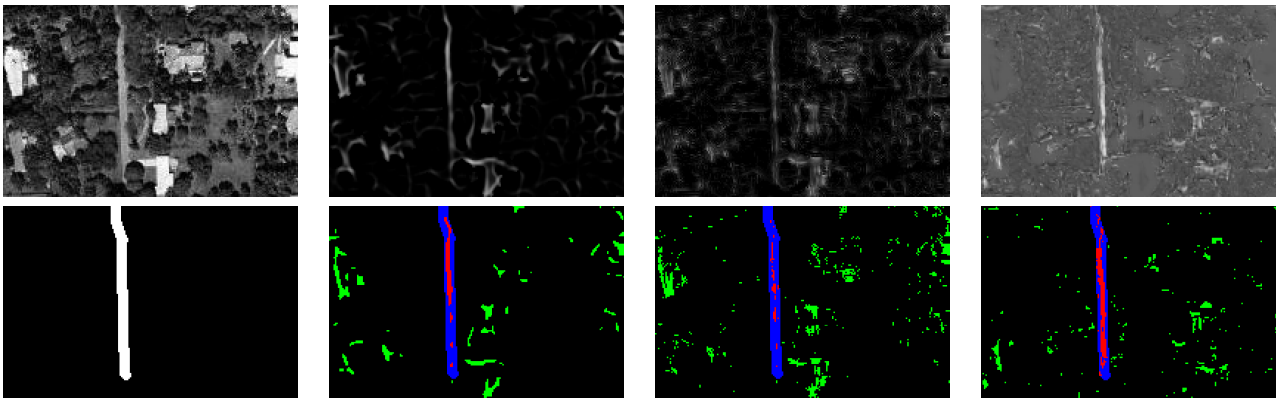
noise produced by man-made structures such as houses or parking lots, which RTF has learned to reject. This robustness to structured noise is demonstrated by results on the close-up of Fig. 7, which contains several white houses. While MVEF and OOF react strongly to them, RTF rejects them almost completely.

4.4 Computational Cost

The ability of our algorithm to learn the irregularities of the data and to handle noise comes at a computational cost. In the $2D$ case, it is approximately 10 times slower than MVEF [10] and OOF [20] when estimating the orientation at each pixel as in Eq. 12 and 100 times slower when testing for all possible orientation. In the $3D$ case, the slowdown factor is closer to 60 when estimating the orientation at each pixel due to the many more convolutions required, the higher complexity of the rotation matrices of Eq. 8, and the larger number of support vectors retained after training the SVM of Eq. 10.

In practice, this means that processing each image of the DRIVE dataset takes approximately 1.5 minutes as opposed to 1 second using OOF on a modern 16-core 2.4 Ghz workstation using parallelized code. Similarly, processing a bright-field image stack of size $384 \times 896 \times 37$ takes about 170 minutes as opposed to only 3.

While the increased computation time is clearly undesirable, our method therefore remains practical because ridge detection is usually performed as a pre-processing step, which can run overnight. Furthermore, the bulk of the computation goes into convolutions with separable filters and the SVM computation, which could be greatly sped up by being reimplemented on a GPU as opposed to a CPU.



Original & GT MVEF [10] OOF [20] RTF

Fig. 7: Detail of the detection on a layer of the STREETS dataset for an FPR of 3%. For such FPR, our algorithm recovers the whole street, while MVEF and OOF recover a much smaller portion. This is due to their high response to the structured noise produced by the nearby houses and parking lots.

5 Conclusions

We have presented an algorithm for detecting ridge-like structures in complicated imagery. Instead of explicitly modeling the filaments and the noise present in the image, we learn a ridge model from the data itself. By so doing, we were able not only to detect filaments, but also to reject structured noise. Our approach was also able to detect non-ideal filament structures, which cannot be easily explicitly modeled, such as junctions or filaments of non-uniform width.

Our main contribution was to combine a machine learning approach with the decomposition of the image into a multi-scale rotational basis to achieve rotational invariance. This allowed us to train a single classifier that learns on data rotated to a canonical orientation. Our classifier was able to detect filaments at any orientation by applying a simple rotation transformation to the extracted feature vector of the pixel under analysis, therefore avoiding expensive orientation dependent convolutions.

We have shown the versatility of our approach by evaluating it on five different types of data: blood vessels in retinal images, neurons in $2D$ and $3D$ bright-field and confocal microscopy and streets in satellite imagery. Our results demonstrate that our approach outperforms state-of-the-art methods in all scenarios, and that the margin of improvement increases with the complexity of the image. The proposed approach is very generic because, instead of postulating *a priori* models for the filaments we are looking for, our algorithm can learn specific appearance models for each new situation.

The code is publicly available [15] and relies on the ITK [17] generic classes. The most computationally demanding part of our processing pipeline is the computation of the filter responses. However, since they are Gaussian and separable, this can be done effectively on a CPU and even more effectively on a GPU. Future work will therefore focus on developing a GPU implementation that can handle the very large amounts of 3D data that modern microscopes now routinely produce.

References

[1] G. Agam and C. Wu. Probabilistic Modeling-Based Vessel Enhancement in Thoracic CT Scans. In *Conference on Computer Vision and Pattern Recognition*, pages 684–689, 2005.

[2] F. Aguet, M. Jacob, and M. Unser. Three-Dimensional Feature Detection Using Optimal Steerable Filters. In *International Conference on Image Processing*, September 2005.

[3] K. Al-Kofahi, A. Can, S. Lasek, D. Szarowski, N. Dowell-Mesfin, W. Shain, J. N. Turner, and et al. Median-Based Robust Algorithms for Tracing Neurons from Noisy Confocal Microscope Images, December 2003.

[4] K. A. Al-Kofahi, S. Lasek, D. H. Szarowski, C. J. Pace, G. Nagy, J. N. Turner, and B. Roysam. Rapid Automated Three-Dimensional Tracing of Neurons from Confocal Image Stacks. *Transactions on Information Technology in Biomedicine*, 6(2):171–187, 2002.

[5] G. A. Ascoli, K. Svoboda, and Y. Liu. Digital Reconstruction of Axonal and Dendritic Morphology Diadem Challenge, 2010. http://diademchallenge.org/.

[6] M. Bioscience. Neurolucida: Microscope Systems for Stereology and Neuron Morphology, 2009. http://www.mbfbioscience.com/neurolucida/.

[7] J. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), 1986.

[8] C. Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.

[9] F. Fleuret and D. Geman. Stationary Features and Cat Detection. *Journal of Machine Learning Research*, 9:2549–2578, 2008.

[10] A. F. Frangi, W. J. Niessen, K. L. Vincken, and M. A. Viergever. Multiscale Vessel Enhancement Filtering. *Lecture Notes in Computer Science*, 1496:130–137, 1998.

[11] W. Freeman and E. Adelson. The Design and Use of Steerable Filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:891–906, 1991.

[12] D. Geman and B. Jedynak. An active testing model for tracking roads in satellite images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1):1 –14, jan 1996.

[13] V. I. GmbH. Amira, 2011. http://www.amira.com.

[14] G. Gonzalez, F. Aguet, F. Fleuret, M. Unser, and P. Fua. Steerable Features for Statistical 3D Dendrite Detection. In *Conference on Medical Image Computing and Computer Assisted Intervention*, pages 625–32, September 2009.

[15] G. Gonzalez, F. Benmansour, E. Turetken, and P. Fua. Rotational Filters - http://cvlab.epfl.ch/software/rtf, 2011. The link will be activated when the paper is accepted.

[16] G. Gonzalez, F. Fleuret, and P. Fua. Learning Rotational Features for Filament Detection. In *Conference on Computer Vision and Pattern Recognition*, pages 1582–1589, June 2009.

[17] L. Ibanez, W. Schroeder, L. Ng, and J. Cates. *The Itk Software Guide. http://www.itk.org/ItkSoftwareGuide.pdf*, 2005.

[18] M. Jacob and M. Unser. Design of Steerable Filters for Feature Detection Using Canny-Like Criteria. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1007–1019, August 2004.

[19] C. Kirbas and F. Quek. Vessel Extraction Techniques and Algorithms: a Survey. In *Symposium on BioInformatics and BioEngineering*, page 238, 2003.

[20] M. Law and A. Chung. Three Dimensional Curvilinear Structure Detection Using Optimally Oriented Flux. In *European Conference on Computer Vision*, pages 368–382, 2008.

[21] D. Lesage, E. D. Angelini, I. Bloch, and G. Funka-Lea. A review of 3D vessel lumen segmentation techniques: Models, features and extraction schemes. *Medical Image Analysis*, 13(6):819–845, 2009.

[22] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[23] E. Meijering, M. Jacob, J.-C. Sarria, P. Steiner, H. Hirling, and M. Unser. Design and Validation of a Tool for Neurite Tracing and Analysis in Fluorescence Microscopy Images. *Cytometry*, 58A(2):167–176, April 2004.

[24] M. Petrou. Optimal Convolution Filters and an Algorithm for the Detection of Wide Linear Features. In *IEE Proceedings I, Vision, Signal and Image Processing*, pages 331–339, 1993.

[25] A. Santamaría-Pang, C. M. Colbert, P. Saggau, and I. Kakadiaris. Automatic Centerline Extraction of Irregular Tubular Structures Using Probability Volumes from Multiphoton Imaging. In *Conference on Medical Image Computing and Computer Assisted Intervention*, pages 486–494, 2007.

[26] Y. Sato, S. Nakajima, H. Atsumi, T. Koller, G. Gerig, S. Yoshida, and R. Kikinis. 3D Multi-Scale Line Filter for Segmentation and Visualization of Curvilinear Structures in Medical Images. *Medical Image Analysis*, 2:143–168, June 1998.

[27] S. Schmitt, J.-F. Evers, C. Duch, M. Scholz, and K. Obermayer. New Methods for the Computer-Assisted 3D Reconstruction of Neurons from Confocal Image Stacks. *NeuroImage*, 23:1283–1298, 2004.

[28] R. Socher, A. Barbu, and D. Comaniciu. A Learning-based Hierarchical Model for Vessel Segmentation. In *IEEE Symposium on Biomedical Imaging: From Nano to Macro*, 2008.

[29] J. Staal, M. Abramoff, M. Niemeijer, M. Viergever, and B. van Ginneken. Ridge Based Vessel Segmentation in Color Images of the Retina. *IEEE Transactions on Medical Imaging*, 23:501–509, 2004.

[30] G. Streekstra and J. van Pelt. Analysis of Tubular Structures in Three-Dimensional Confocal Images. *Network: Computation in Neural Systems*, 13(3):381–395, August 2002.

[31] E. Turetken, G. González, C. Blum, and P. Fua. Automated reconstruction of dendritic and axonal trees by global optimization with geometric priors. *Neuroinformatics*, 9:279–302, May 2011.

[32] J. Tyrrell, E. di Tomaso, D. Fuja, R. Tong, K. Kozak, R. Jain, and B. Roysam. Robust 3D Modeling of Vasculature Imagery Using Superellipsoids. *Medical Imaging*, 26(2):223–237, February 2007.

[33] Y. Wang, A. Narayanaswamy, C. Tsai, and B. Roysam. A broadly applicable 3-d neuron tracing method based on open-curve snake. *Neuroinformatics*, 9(2-3):193–217, 2011.

APPENDIX: ROTATION EQUATIONS

For completeness sake, we re-derive here the coefficients of the rotation matrices we use to steer the filters using the same formalism as in [18]. We do this first in $2D$ and then in $3D$.

A 2D STEERING EQUATIONS

A 2D steerable filter based on Gaussian derivatives can be written as

$$h(\mathbf{u}) = \langle \alpha, \mathbf{f} \rangle = \sum_{q=1}^{M} \sum_{m=0}^{q} \alpha_{m,q} f_{m,q}, \quad f_{m,q} = \frac{\partial^m}{\partial x^m} \frac{\partial^{q-m}}{\partial y^{q-m}} g(\mathbf{u}) \ , \tag{14}$$

where the α vector controls the shape of the filter. Computing spatial derivatives is equivalent to multiplying by the frequency in the Fourier domain. The Fourier transform of Eq. 14 is

$$\hat{h}(\mathbf{w}) = \sum_{q=1}^{M} \sum_{m=0}^{q} \alpha_{m,q} (jw_x)^m (jw_y)^{q-m} \hat{g}(\mathbf{w}) \ , \tag{15}$$

which is a polynomial that multiplies a radially symmetric function. Since polynomials are steerable and a rotation in the Fourier domain corresponds to a rotation in the space domain, by Theorem 3 of [11], we know that the filter is steerable.

Making the change of variable $\mathbf{w}' = (R^\theta)^T \mathbf{w}$ and expanding yields

$$\begin{aligned} \hat{h}(R^\theta \mathbf{w}) &= \sum_{q=1}^{M} \sum_{m=0}^{q} \alpha_{m,q} (jw_x \cos(\theta) + jw_y \sin(\theta))^m \\ &\quad (-jw_x \sin(\theta) + jw_y \cos(\theta))^{q-m} \hat{g}(\mathbf{w}) \ , \\ &= \sum_{q=1}^{M} \sum_{m=0}^{q} \alpha_{m,q} \sum_{i=0}^{m} \sum_{k=0}^{q-m} \frac{m!(q-m)!}{i!(m-i)!k!(q-m-k)!} \\ &\quad (-1)^k (jw_x)^{k+i} (jw_y)^{q-k-i} \cos(\theta)^{q-m-k+i} \sin(\theta)^{k+m-i} \hat{g}(\mathbf{w}) \ , \end{aligned}$$

Computing the inverse Fourier transform yields

$$\begin{aligned} h(R^\theta \mathbf{u}) &= \sum_{q=1}^{M} \sum_{m=0}^{q} \alpha_{m,q} \ , \\ &\quad \sum_{i=0}^{m} \sum_{k=0}^{q-m} \frac{m!(q-m)!}{i!(m-i)!k!(q-m-k)!} (-1)^k \cos(\theta)^{q-m-k+i} \sin(\theta)^{k+m-i} f_{k+i,q} \ . \end{aligned} \tag{16}$$

Let $\mathcal{S}(q,m,j)$ be the set $\{i,k | 0 \le i \le m; 0 \le k \le q-m; i+k=j\}$. We can rewrite 16 as

$$\begin{aligned} h(R^\theta \mathbf{u}) &= \sum_{q=1}^{M} \sum_{m=0}^{q} \alpha_{m,q} \\ &\quad \sum_{j=0}^{q} \sum_{i,k \in \mathcal{S}(q,m,j)} \frac{m!(q-m)!}{i!(m-i)!k!(q-m-k)!} (-1)^k \cos(\theta)^{q-m-k+i} \sin(\theta)^{k+m-i} f_{k,j} \ . \end{aligned} \tag{17}$$

Interchanging the summation in m with the summation over j yields

$$\begin{aligned} h(R^\theta \mathbf{u}) &= \sum_{q=1}^{M} \sum_{j=0}^{q} f_{j,q} \\ &\quad \underbrace{\sum_{m=0}^{q} \alpha_{m,q} \sum_{i,k \in \mathcal{S}(q,m,j)} \frac{m!(q-m)!}{i!(m-i)!k!(q-m-k)!} (-1)^k \cos(\theta)^{q-m-k+i} \sin(\theta)^{k+m-i}}_{\beta_{j,q}(\theta)} \end{aligned} \tag{18}$$

This equation can be interpreted as a sparse rotation matrix. To compute $\beta_{j,q}(\theta)$, only the coefficients $\alpha_{m,q}$ are taken into account. Therefore the rotation properties hold for each order independently. It can be rewritten as

$$a^q_{j,m} = \sum_{i,k \in \mathcal{S}(q,m,j)} (-1)^k \cos(\theta)^{q-m-k+i} \sin(\theta)^{k+m-i}, \tag{19}$$

or more compactly as

$$\beta(\theta) = \mathcal{R}^\theta \alpha \ . \tag{20}$$

Finally, the rotated steerable filter is re-written as

$$h(R^\theta \mathbf{u}) = \langle \mathcal{R}^\theta \alpha, f \rangle \ . \tag{21}$$

B 3D Steering Equations

Following the same approach as in the 2D case, a 3D rotational filter based on Gaussian derivatives can be written as

$$h(\mathbf{u}) = \langle \alpha, \mathbf{f} \rangle = \sum_{q=1}^{M} \sum_{m=0}^{q} \sum_{n=0}^{q-m} \alpha_{m,n,q} f_{m,n,q} \ , \tag{22}$$

$$f_{m,n,q} = \frac{\partial^m}{\partial x^m} \frac{\partial^n}{\partial y^n} \frac{\partial^{q-m-n}}{\partial y^{q-m-n}} g(u) \ , \tag{23}$$

and its Fourier transform is

$$\hat{h}(\mathbf{w}) = \sum_{q=1}^{M} \sum_{m=0}^{q} \sum_{n=0}^{q-m} \alpha_{m,n,q} (jw_x)^m (jw_y)^n (jw_z)^{q-m-n} \hat{g}(\mathbf{w}) \ . \tag{24}$$

b.1 3D Rotation

Using Euler angles, a rotation in the 3D space can be written as

$$R = R^\theta_X R^\psi_Y R^\phi_Z \ , \tag{25}$$

where R^θ_A is the rotation of angle θ around axis A, where A stands for either X, Y, or Z. Therefore, computing the steering equations for a generic rotation, amounts to computing those around each individual axis. We show below that, as in the case of the 2D steering equations of Eq. 21, we can write

$$\forall \mathbf{u}, \ h(R^\theta_X \mathbf{u}) = \langle \alpha, \mathbf{f}(R^\theta_X \mathbf{u}) \rangle = \langle \mathcal{R}^\theta_X \alpha, \mathbf{f}(\mathbf{u}) \rangle \ . \tag{26}$$

Since $\mathcal{R}^\psi_Y$ and $\mathcal{R}^\phi_Z$ can be computed similarly, we have

$$h(R\mathbf{u}) = \langle \mathcal{R}\alpha, \mathbf{f}(\mathbf{u}) \rangle, \tag{27}$$

where

$$\mathcal{R} = \mathcal{R}^\theta_X \mathcal{R}^\psi_Y \mathcal{R}^\phi_Z \ . \tag{28}$$

b.2 3D Rotation around the X axis

We now turn to proving Eq. 26 and compute the steering equation for R^θ_X, the rotation of angle θ around the X axis. The steering equation for R^ψ_Y and R^ϕ_Z, the rotations around the other two axes, can be computed similarly.

Making the change of variable $\mathbf{w}' = (R^\theta_X)^{-1}\mathbf{w}$ and using the rotational properties of the Fourier transform yields

$$\begin{aligned} jw'_x &= jw_x \ , \\ jw'_y &= \cos(\theta) jw_y + \sin(\theta) jw_z \ , \\ jw'_z &= -\sin(\theta) jw_y + \cos(\theta) jw_z \ . \end{aligned}$$

Substituting in Eq. 24 and performing some algebraic manipulations yields

$$\begin{aligned}\hat{h}(R_X^\theta \mathbf{w}) &= \sum_{q=1}^{M}\sum_{m=0}^{q}\sum_{n=0}^{q-m}\alpha_{m,n,q}\sum_{i=0}^{n}\sum_{k=0}^{q-m-n}\frac{(q-n-m)!n!(-1)^k}{k!(q-m-n-k)!i!(n-i)!} \\ &\quad \cos(\theta)^{q+i-m-n-k}\sin(\theta)^{k+n-i}(jw_x)^m(jw_y)^{k+i}(jw_z)^{q-m-k-i}\hat{g}(\mathbf{w})\ .\end{aligned} \tag{29}$$

The inverse Fourier transform is

$$\begin{aligned}h(R_X^\theta \mathbf{u}) &= \sum_{q=1}^{M}\sum_{m=0}^{q}\sum_{n=0}^{q-m}\alpha_{m,n,q}\sum_{i=0}^{n}\sum_{k=0}^{q-m-n}\frac{(q-n-m)!n!(-1)^k}{k!(q-m-n-k)!i!(n-i)!} \\ &\quad \cos(\theta)^{q+i-m-n-k}\sin(\theta)^{k+n-i}f_{m,k+i,q}\ .\end{aligned} \tag{30}$$

Let $\mathcal{S}_X(r,s,m,n)$ be the set $\{i,k|0\le i\le n; 0\le k\le q-m-n; m=r; k+i=s\}$. Eq. 30 can be rewritten as

$$\begin{aligned}h(R_X^\theta \mathbf{u}) &= \sum_{q=1}^{M}\sum_{m=0}^{q}\sum_{n=0}^{q-m}\alpha_{m,n,q}\sum_{r=0}^{q}\sum_{s=0}^{q-r}\sum_{i,k\in\mathcal{S}_X(r,s,m,n)} \\ &\quad \frac{(q-n-m)!n!(-1)^k}{k!(q-m-n-k)!i!(n-i)!}\cos(\theta)^{q+i-m-n-k}\sin(\theta)^{k+n-i}f_{r,s,q}\ .\end{aligned}$$

Changing the summation order yields

$$\begin{aligned}h(R_X^\theta \mathbf{u}) &= \sum_{q=1}^{M}\sum_{r=0}^{q}\sum_{s=0}^{q-r}f_{r,s,q} \\ &\quad \underbrace{\sum_{m=0}^{q}\sum_{n=0}^{q-m}\alpha_{m,n,q}\sum_{i,k\in\mathcal{S}_X(r,s,m,n)}\frac{(q-n-m)!n!(-1)^k}{k!(q-m-n-k)!i!(n-i)!}\cos(\theta)^{q+i-m-n-k}\sin(\theta)^{k+n-i}}_{\beta_{r,s,q}(\theta)}\ .\end{aligned} \tag{31}$$

As in the $2D$ case, Eq. 31 can be written in matrix form as

$$a_{m,n}^{r,s,q} = \sum_{i,k\in\mathcal{S}_X(r,s,m,n)}\frac{(q-n-m)!n!(-1)^k}{k!(q-m-n-k)!i!(n-i)!}\cos(\theta)^{q+i-m-n-k}\sin(\theta)^{k+n-i}\ , \tag{32}$$

or, more compactly, as

$$\beta_X(\theta) = \mathcal{R}_X^\theta \alpha\ . \tag{33}$$