Paper…. Think of a funky title here…

Introduction

* What structures?
  + Curvi-linear structures
* Why are they interesting?
* What are their general characteristics?
  + Various amount of stuff in image
  + Can appear anywhere
  + Defined locally by their cross-sectional profile and orientation, and assumed to extend in at least one direction normal to the profile (as opposed to a blob). Assumed to have finite width, and thus have a profile that can approximately be described as symmetrical
* We explore methods for predicting, at every pixel in an image, the probability that the pixel is a part of a CLS and the local attributes of the CLS (orientation, width etc). We describe these as low-level local attributes
* These attributes can be used in a vast number of different ways, from further low-level and mid-level processing to direct uses in high-level applications.
* We list some of these as motivation for our work…
  + Further tasks…
* However our goal is not to propose novel methods for any such further processing. Instead, our focus is on computing the local attributes as accurately and robustly as possible, on the assumption that this can only be of benefit to whichever methods later use this information.
* Why make this separation? Because we believe the information we compute in this work is generically useful over a large class of images and applications. In contrast, higher level processing is likely to benefit from application specific methods that make assumptions unique to that problem.
  + For example, any method to move from a map of vessel probabilities and predicted orientations in a retinograms, to an explicit grouping of pixels that belong to individual vessels, is likely to benefit from a priori knowledge of the spatial arrangement vessels in that image class and the physical model of how vessels grow and bifurcate. Such a method will therefore be very different from that needed to group a similar set of local information into the road, rivers etc present in an image for aerial analysis
* By restricting our focus in this way, we can concentrate on systematically exploring the two key components of the task, namely 1) how to efficiently extract sufficient structural information at each image location and 2) how to combine this raw local information to predict outputs useful to further work.
* For the former, we consider sets of image filter banks used commonly in previous literature. Whilst we don’t claim to exhaustively test all filters previously used, we aim to show the relationship between the most common in the context of what local information can be captured from the image. In doing so, we describe the necessary properties of a suitable filter bank and can then make a principled choice that compromises between the richness of information extracted and efficiency (both computational and storage). In all cases, we back up theoretical claims for filter performance with thorough experimental validation on both synthetic and real data.
* For the latter task, we propose a supervised learning approach, in which the filter responses at a given location are pooled into a feature vector. Given a suitable set of training examples consisting of feature vector/output measure pairs, we can train a classifier/regressor to predict unseen test data using standard machine learning algorithms.
* Such an approach has most commonly been used in the task of segmenting structure – i.e., in the context of our definitions above, predicting a probability of each pixel belonging to structure or background. We add to this by showing how careful, principled selection of the filter bank, coupled with a modern classifier/regressor capable of dealing with highly complex non-linear data, can produce segmentations that match or exceed the state-of-the-art, without relying on application specific assumptions to boost the performance.
  + Although we should probably acknowledge that we do in fact gain a large amount of image specific knowledge in terms of the labels assigned to our training data – but this is the only knowledge we seek to exploit.
* However, we further propose that supervised learning may be applied to predict the other local attributes of interest such a structure orientation and width. In this regard, bar some initial orientation results we have previously published, our work is unique. For orientation, we show how to formulate the learning problem such that angle wraparound is dealt with correctly, and in doing so produce significantly more accurate estimations than the methods used to compute orientation analytically in the vast majority of earlier work. In addition, we show how predicting output responses via machine learning allows us to estimate the error associated with the predictions, which we propose in itself may be of us in further processing.
* Finally, if for a given image class we know of additional properties that will be useful to predict and that can be reliably labelled on our training data (for example, a further classification of structure in aerial images to roads and rivers) these can easily be added to our protocol.
* Summary of contributions:
  + Detailed analysis of the relationships between commonly used filter banks and their suitability for extracting local information relevant to CLS from images
  + Comparison of filter bank performance for both linear and non-linear machine learners in both synthetic and real data
  + “Joined up” approach that uses the same feature vectors for low-level segmentation to predict other useful local attributes such as orientation and width. Where modifications to standard machine learning are required (e.g. to deal correctly with angle wraparound when predicting orientation) these are explicitly stated.
  + State-of-the-art performance, with particularly significant improvement in orientation estimation over traditional analytical methods.
* What’s in the rest of paper:
  + Literature review
    - * Filtering basics
      * Line detection/classification papers
      * Machine learning?
  + Filters
  + Forming feature vectors
  + Output measures
  + Machine learning
  + Experiments
    - * Synthetic
      * Retinograms
      * Google maps? Nailfolds?
  + Discussion
  + Conclusions

Choosing filters….

In this section we consider the theoretical requirements of a suitable filter bank, keeping foremost in our mind the application in which the responses will be used. That is the set of responses for any given structure pixel should be distinguishable from that of any background pixel; the filters should be directionally selective to predict orientation [[1]](#footnote-1); likewise to predict structure width the responses should be selective across scale.

We start by considering probably the most commonly used set of filters for linear structure detection (certainly within the context of vessel segmentation in medical images): derivatives of a Gaussian kernel.

Gaussian first derivatives – edge detection. Explain second derivatives. Note this approach is often reformulated by creating a Hessian matrix with the xx and yy derivates on the lead diagonal and the xy derivatives on the opposing diagonal. Solving this matrix produces eigen vectors with direction theta and thetaN and responses R.

G2D’s work on the assumption that when steered to match the orientation of a structure, the response will be large, whilst the response in the perpendicular direction will be near-zero. Thus structures can be distinguished from flat backgrounds (both responses near-zero) or circular blob-like structures (both responses large). Equation () (or its Hessian reformulation) provides an elegant solution for determining this orientation analytically, from which the responses can be used to detect structure directly []. Alternatively the filters may be steered to multiple orientations over multiple scales and used as features in a machine learning algorithm [].

However, there are two problems in using second derivatives alone. Firstly, a strong edge in the image will produce an “echo” response that cannot be distinguished from the response at the centre of a CLS. This may seem an arbitrary construct; however it is easy find examples in real data. For example, the edge of the optic disc in a retinogram (Fig X) is often misclassified as a vessel, whilst similar artefacts may occur at the edges of lesions or near the pectoral muscle in mammograms.

Secondly, due to noise in the image, the symmetric profile of a CLS may be disrupted to the extent that not only does the equation [] produce inaccurate results, the responses themselves hold no usual information that a machine learning algorithm to take advantage of. Again, this can be seen clearly in real data, particularly in structures that have a width of only one or two pixels, such as the smallest vessels in retinograms.

The first problem tells us that it is not enough to only have filters designed to match the shape profile of the CLS we want to detect if such filters cannot distinguish other structures in the image background. The second problem shows we cannot rely on the assumptions we make about the CLS in real images. Combining both ideas motivates us to choose a filter bank that more generally represents an image. Our goal then is not to make a priori assumptions about how we want filters to respond to particular structures, but simply to ensure the set of filter responses produce a unique signature for differing structures. It is then up to our chosen machine learning algorithm to match the various signatures present in the training data to the output measure of interest.

Returning to Gaussian derivatives, we note the cause of both problems is that at a given scale and orientation, we can only compute the response to a filter with even symmetry. An intuitive solution is to supplement these filters with Gaussian 1st derivatives (as used most commonly in edge detection e.g. Canny) as used to heuristically discard edge echo responses in []. However, we prefer the solution recommended in [], using the Hilbert transform of the second derivatives. A steerable response can be computed from four separable basis filters, defined below:

The advantage of using R\_h rover first derivatives is that it allows us to represent the responses as magnitude and phase:

We show in section X how this improves results, particularly for orientation (and width?) prediction.

Next we consider Gabor filters. Equations:

As with Gaussian derivatives, these have been used extensively in detecting CLS, although again often only the even filter [eq X] is used as this is what is assumed will match the shape. For the same reason we match G2 with its Hilbert pair, we recommend using both odd and even parts, with maximum benefits obtained by combining them as a magnitude/phase pair and show experimentally the advantages of doing so in section X.

Note that unlike Gaussian derivatives, Gabor filters are neither separable nor steerable, making them much more expensive to compute. In contrast we now consider two further filtering schemes designed to measure magnitude and phase across and orientation and scale more efficiently.

The monogenic signal description…

The DT-CWT description…

Finally we acknowledge that there are of course many further filter banks we do not test in this paper, and for which an exhaustive comparison of results is unfeasible. However, we show that it is the properties we choose to implement for a given a filter bank (e.g. a magnitude/phase versus just an even/odd response, decimation versus increasing filter size, oversampling scales etc.) rather than the inherent properties of the filters that have the biggest effect in performance. In turn, given a set computational cost, this allows us to make an informed choice of suitable filter bank for any given data.

We also show that a filter bank selected given these general criteria can produce features that outperform features handcrafted for a particular application.

One interesting concept we do not test is the idea of learning an optimal set of arbitrary filters for a given set of images, as in []. However, we believe that such an approach is only beneficial if the filters are optimised with respect to the task they need to perform (in our case and in [], separating the responses for CLS and background pixels within a classifier) and cannot see how optimising with respect to some other task (such as reconstructing the image in a maximally sparse way) is intrinsically a desirable thing to do. That said a comparison with the results in [] would be desirable if quantitative results on the DRIVE and STARE datasets were made available.

In this section we consider how, for any pixel, to combine the responses of a given filter bank into a feature vector. In the simplest form, we just concatenate the responses from the raw filters at all scales and orientations, however we also consider the following:

Steering:

For the Gaussian derivatives (and their Hilbert transform), we can choose to steer the raw responses at each scale to a fixed set of directions spread evenly across the circle (e.g. in the same directions we apply the Gabor filterbank). This potentially produces features that can be more easily matched to the appropriate output measure and for experimentation purposes provides a more direct comparison to the Gabor and DT-CWT filter banks.

Complex form:

As discussed in the previous section, for the Gaussian, Gabor and DT-CWT filter banks at a given scale and orientation we have a pair of responses that can be thought of as a complex number, where by custom we use the response to the even filter for the real part and the odd response for its imaginary counterpart. We can then choose either to include the real and imaginary parts as separate dimensions in the feature vectors or represent the pair of responses as magnitude and phase. The latter is arguably a more intuitive way to think of the responses – considering the image profile sampled at a given orientation and scale, the magnitude signifies if a feature is present in this 1D signal, whilst the phase tells us about the shape of the feature as it varies from a valley, to a step, to a ridge. Note also that if an image is rotated through 180 degrees, a complex response C = a+ib becomes C\*=a-ib. Thus if we want to make our features responses ambivalent to 180 degree rotations we can use the absolute value of the imaginary part when computing phase.

Rotation invariance

Given a set of responses at orientations spread evenly over the circle, a common approach is to select the orientation with maximal response and circular shift the remaining responses in the feature vector such that the maximal orientation for each feature vector occupies the same dimension. The intention is to produce feature vectors with rotational invariance thus collapsing the size of the feature space (proportional to the number of discrete orientations in the filter bank) and making it easier for the classifier to do its job. Whilst theoretically appealing we show in section X that this has no discernible benefit in practice. Note also, with responses over multiple scales, we can choose either to allow the responses in each scale to shift independently or choose a single maximum orientation (e.g. from the scale that produces maximum response) to circularly shift all scales. Here we take the former approach although we have experimented with both and found little difference in performance.

Pooling neighbourhood responses

In contrast to faffing with rotational invariance, a simple yet effective measure is to pool the responses from neighbouring pixels. Again we have experimented with more exotic sampling schemes in which we interpolate responses in a circular pattern about the pixel of interest, but have found that in practice simply sampling a 3x3 window of responses provides most benefit. Of course this increases the dimension of the feature vectors nine-fold, but with machine learning algorithms (such as random forests) designed to cope with large dimensional features and plenty of training data (which is nearly always the case in this set up given each pixel in an image is a sample and thus even a small set of images typically contain millions of samples) this needn’t be a problem.

Flow chart…

**Outcome measures**

About outcome measures

**Machine Learning**

About random forests…

**Sampling data**

The final step in our method is to determine how we sample data for the forests. We have two schemes, one for running experiments on training data (e.g. to evaluate parameter options), the other for making final predictions on test data.

In the first scheme, we simply take some fixed size random subsample of pixels across the whole training data, with an equal number of background and foreground pixels (although only the foreground pixels are used orientation and width prediction). We then take a bootstrap sample of this data to train each tree in the forest. To test the forest, we take a second subsample from the pixels in the main training data not used in the first set. We can repeat this scheme, taking different random subsamples at every iteration, to compute a measure of uncertainty in prediction performance.

To make final predictions on the test data, we adopt a slightly more complicated sampling scheme that aims to better use all the data in the training set. In the first stage, we sample a different random subset of the training data for each tree during forest building, recording which pixels were selected. We then use the forest to predict all the training data, where at each pixel we aggregate only those predictions from trees for which the pixel wasn’t selected. We can thus produce an unbiased prediction error at each pixel, analogous to the out-of-bag error described in Breiman’s original random forest work [].

We then build a second forest, where again we sample a different random subset of the training data for each tree, only this time rather than uniformly sampling from the data, we weight the samples according to equation X,

X = Y

where Ep is the prediction error described above and is defined separately for detection, orientation and width prediction as follows:

X=a

Y =b

Z=c

This has the effect of oversampling pixels that were poorly predicted in the first forest (with lambda controlling the level of oversampling versus uniform sampling) and results in a significant improvement in overall prediction performance. Note that the second stage of this process can be repeated to determine a suitable value for lambda. Indeed subject to time constraints, we could iterate until our predictions in the training data converge. In practice however, we evaluate performance for a fixed set of values for lambda and select the best.

The resulting forest can then be used to make predictions for all images in the test data.

**Data**

We evaluate our methods on 3 sets of real data:

DRIVE (description)

STARE (description)

Nerve fibres (description)

In addition, we use synthetic data to explicate particular points discussed in section X. Each synthetic image is created as follows:

Generate a line of random direction, contrast and width, with elliptical profile centred in 64x64 background. The background may either be flat or containing an edge of random orientation and contrast. The image is then corrupted by signal dependent Rician noise, subject to equation X.

X = y

**Experiments**

We first show use sets of synthetic data with increasing noise to highlight the benefit of learning to predict orientation as opposed to relying on analytical methods (we take it as given now that learning is established as superior to analytical methods for detecting structure), and to examine the benefit of using both odd and even filters.

We then test all combinations of composing feature vectors from our four filter banks using subsamples of the training data for each real dataset.

Finally, we use the best performing feature vector composition and construct forests to apply to the test data for the DRIVE (and STARE?) data as described in section X, and compare the results to previous work.

Experiment 1 – synthetic data, increasing noise

Figures:

* Percentage of correctly selected oriented sub-band as noise increases
* Percentage of correctly selected scale as noise increases
* Overall Gaussian prediction error
* The same for DRIVE data
* Performance for individual scales

\*Key point: analytic solutions don’t effectively compile information of all scale – you are better off picking a single scale. In contrast, learning methods improve as all scales are included. Note also that including all scales provides a more consistent performance over structures of all widths\*

Experiment 2 (may leave this out?) – synthetic data, line on edge

* Line detection when on edge: with/without odd component, 1x1 vs 3x3
* Performance specifically at edge:
  + Show figure on synthetic data
  + Real example from DRIVE data

Experiment 3 – real data, all permutations of composing feature vectors

See powerpoint results slides

Key points:

* Overall:
  + Detection: Gabor = Gaussian > DT-CWT > Monogenic
  + Orientation: Gabor > DT-CWT > Gaussian > Monogenic
  + Hypothesise that this is due to Gabor being more directionally selective than Gaussian (see figure X)
* Steering:
  + Yes, this benefits Gaussian coefficients
* Including odd/even filters
  + Yes, benefits Gabor and Gaussian filters (for DT-CWT it is particularly necessary as neither filter is completely odd or even, thus neither is an ideal match for the majority of data)
  + Benefit particularly noticeable in predicting orientation
  + Converting to magnitude/phase further benefit orientation prediction
  + Conjugate phase also improves orientation prediction
* Rotation invariance
  + No significant benefit for Gaussian and Gabor
  + Significantly worse for DT-CWT (because bands are not rotationally identical see figure X)
* Pooling neighbourhood responses
  + Always benefits, regardless of test
  + Allows filters with odd symmetry to approximate performance of filters with even symmetry
  + Increases size of feature vector (and hence tree training and predicting time) but doesn’t include additional filtering overhead
* Pooling over all scales
  + Always benefits, regardless of test
  + Produces consistent prediction across structures of all size: using just filters from “middle scales” may give similar performance averaged over the whole set, but performs significantly worse for fine structures
* Oversampling scale and orientation
  + Has benefit, though tiny margins for detection
  + Scale more beneficial than orientation
* Putting all filters together
  + Benefit, but less so than oversampling scale (see above) which is cheaper and requires less memory
* Comparing datasets
  + STARE and DRIVE very similar
  + As expected, FIBRE harder than both

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Retinograms: DRIVE data | | | | | |
|  | | Vessel Detection | | Vessel Orientation | |
| *w = 1* | *w = 3* | *w = 1* | *w = 3* |
| Gaussian | *G* | 0.947± 6.4 | 0.959 ± 5.3 | 6.89 ± 0.042 | 6.04 ± 0.037 |
| *H* | 0.838 ± 11.0 | 0.957 ± 6.4 | 10.7 ± 0.046 | 6.62 ± 0.031 |
| *G + H* |  |  |  | 5.96 ± 0.028 |
| Gaussian, steered to 6 directions | *G* | 0.949 ± 5.0 | 0.961 ± 5.5\* | 6.71 ± 0.038 | 5.91 ± 0.022 |
| *H* |  |  |  | 6.55 ± 0.029 |
| *G + H* |  | 0.962 ± 5.8\* |  | 5.85 ± 0.026 |
| Gabor, 6 directions | *Re* |  | 0.960 ± 5.7\* |  | 5.47 ± 0.021 |
| *Im* |  | 0.953 ± 6.5 |  | 5.52 ± 0.024 |
| *Re + Im* |  | 0.962 ± 5.7\* |  | 5.34 ± 0.019 |
| DT-CWT | *Re* |  | 0.951 ± 6.1\* |  | 5.78 ± 0.023 |
| *Im* |  | 0.951 ± 6.6\* |  | 5.70 ± 0.028 |
| *Re + Im* |  | 0.956 ± 5.9\* |  | 5.51 ± 0.027 |

|  |  |  |  |
| --- | --- | --- | --- |
| Retinograms: DRIVE data | | | |
|  | | Vessel Detection  *(w = 3)* | Vessel Orientation  *(w = 3)* |
| Gaussian, steered to 6 directions | *Mag* | 0.932 ± 7.6 | 5.82 ± 0.023 |
| *Phase* | 0.925 ± 6.5 | 6.50 ± 0.043 |
| *Mag + Phase* |  |  |
| *… using |Im|* | 0.962 ± 4.7 | 5.68 ± 0.020 |
| *… rotated* | 0.960 ± 5.4 | --- |
| *… interpolated* | 0.959 ± 4.6 | 6.06 ± 0.034 |
| *… additional scales* |  |  |
| *… additional directions* |  |  |
| Gabor, 6 directions | *Mag* | 0.932 ± 8.6 | 4.92 ± 0.023 |
| *Phase* | 0.912 ± 8.6 | 9.38 ± 0.059 |
| *Mag + Phase* | 0.959 ± 5.9 | 4.93 ± 0.015 |
| *… using |Im|* | 0.962 ± 6.6 | 4.85 ± 0.020 |
| *… rotated* | 0.963 ± 4.7 | --- |
| *… interpolated* | 0.959 ± 5.0 | 5.44 ± 0.022 |
| *… additional scales* | 0.963 ± 5.5 | 4.59 ± 0.017 |
| *… additional directions* | 0.963 ± 5.8 | 4.74 ± 0.021 |
| DT-CWT | *Mag* | 0.918 ± 7.2 | 5.05 ± 0.02 |
| *Phase* | 0.879 ± 7.9 | 10.8 ± 0.063 |
| *Mag + Phase* | 0.955 ± 5.9 | 4.98 ± 0.026 |
| *… using |Im|* | 0.953 ± 5.9 | 4.96 ± 0.024 |
| *… rotated* | 0.946 ± 6.0 | --- |
| Monogenic | | 0.951 ± 5.5 | 7.85 ± 0.045 |
| All filters | | 0.964 ± 5.0 | 4.63 ± 0.024 |

|  |  |  |  |
| --- | --- | --- | --- |
| Retinograms: DRIVE data | | | |
|  | | Vessel Detection  *(w = 3)* | Vessel Orientation  *(w = 3)* |
| Gaussian, steered to 6 directions | *1* | 0.941 ± 0.00070 | 6.61 ± 0.030 |
| *2* | 0.949 ± 0.00063 | 5.87 ± 0.032 |
| *4* | 0.916 ± 0.00049 | 7.51 ± 0.037 |
| *8* | 0.820 ± 0.00110 | 13.8 ± 0.055 |
| *16* | 0.727 ± 0.00140 | 20.4 ± 0.090 |
| Gabor, 6 directions | *1* | 0.921 ± 0.00095 | 9.27 ± 0.033 |
| *2* | 0.952 ± 0.00069 | 5.89 ± 0.022 |
| *4* | 0.946 ± 0.00047 | 5.00 ± 0.015 |
| *8* | 0.900 ± 0.00045 | 7.09 ± 0.040 |
| *16* | 0.833 ± 0.00083 | 12.2 ± 0.070 |
| DT-CWT | *1* | 0.786 ± 0.00110 | 11.91 ± 0.074 |
| *2* | 0.916 ± 0.00091 | 5.93 ± 0.029 |
| *4* | 0.916 ± 0.00051 | 5.60 ± 0.018 |
| *8* | 0.857 ± 0.00079 | 10.00 ± 0.038 |
| *16* | 0.845 ± 0.00130 | 12.10 ± 0.076 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Retinograms: STARE data | | | | | |
|  | | Vessel Detection | | Vessel Orientation | |
| *w = 1* | *w = 3* | *w = 1* | *w = 3* |
| Gaussian | *G* | 0.946 ± 6.2 | 0.959 ± 6.0 | 6.91 ± 0.049 | 6.06 ± 0.041 |
| *H* |  |  | 10.7 ± 0.043 | 6.64 ± 0.030 |
| *G + H* |  |  |  |  |
| Gaussian, steered to 6 directions | *G* |  |  | 6.72 ± 0.041 | 5.93 ± 0.023 |
| *H* |  |  |  |  |
| *G + H* |  |  |  |  |
| Gabor, 6 directions | *Re* |  |  |  |  |
| *Im* |  |  |  |  |
| *Re + Im* |  |  |  |  |
| DT-CWT | *Re* |  |  |  | 5.80 ± 0.022 |
| *Im* |  |  |  | 5.73 ± 0.026 |
| *Re + Im* |  |  |  | 5.54 ± 0.028 |

|  |  |  |  |
| --- | --- | --- | --- |
| Retinograms: STARE data | | | |
|  | | Vessel Detection  *(w = 3)* | Vessel Orientation  *(w = 3)* |
| Gaussian, steered to 6 directions | *Mag* |  |  |
| *Phase* |  |  |
| *Mag + Phase* |  |  |
| *… using |Im|* |  |  |
| *… rotated* |  |  |
| *… interpolated* | 0.959 ± 5.9 | 6.08 ± 0.038 |
| *… additional scales* |  |  |
| *… additional directions* |  |  |
| Gabor, 6 directions | *Mag* |  | 4.93 ± 0.022 |
| *Phase* |  | 9.43 ± 0.049 |
| *Mag + Phase* |  | 4.94 ± 0.017 |
| *… using |Im|* |  | 4.87 ± 0.017 |
| *… rotated* | 0.962 ± 4.3 |  |
| *… interpolated* | 0.958 ± 6.0 | 5.46 ± 0.026 |
| *… additional scales* |  |  |
| *… additional directions* |  |  |
| DT-CWT | *Mag* |  | 5.08 ± 0.027 |
| *Phase* |  | 10.80 ± 0.070 |
| *Mag + Phase* |  | 5.00 ± 0.022 |
| *… using |Im|* |  | 4.99 ± 0.029 |
| *… rotated* | 0.945 ± 5.7 |  |
| Monogenic | |  |  |
| All filters | |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Retinograms: STARE data | | | |
|  | | Vessel Detection  *(w = 3)* | Vessel Orientation  *(w = 3)* |
| Gaussian, steered to 6 directions | *1* |  |  |
| *2* |  |  |
| *4* |  |  |
| *8* |  |  |
| *16* |  |  |
| Gabor, 6 directions | *1* | 0.920 ± 0.00097 | 9.32 ± 0.032 |
| *2* | 0.952 ± 0.00072 | 5.90 ± 0.019 |
| *4* | 0.945 ± 0.00050 | 5.02 ± 0.016 |
| *8* | 0.899 ± 0.00047 | 7.11 ± 0.036 |
| *16* | 0.831 ± 0.00120 | 12.20 ± 0.055 |
| DT-CWT | *1* | 0.784 ± 0.0012 | 12.00 ± 0.072 |
| *2* | 0.915 ± 0.00097 | 5.94 ± 0.031 |
| *4* | 0.915 ± 0.00054 | 5.62 ± 0.021 |
| *8* | 0.856 ± 0.00076 | 10.00 ± 0.046 |
| *16* | 0.842 ± 0.00130 | 12.20 ± 0.076 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Fibre data | | | | | |
|  | | Vessel Detection | | Vessel Orientation | |
| *w = 1* | *w = 3* | *w = 1* | *w = 3* |
| Gaussian | *G* | 0.896 ± 3.2 | 0.902 ± 3.2 | 8.82 ± 0.16 | 8.43 ± 0.17 |
| *H* | 0.819 ± 1.8 | 0.892 ± 2.7 | 16.60 ± 0.47 | 8.98 ± 0.24 |
| *G + H* |  | 0.903 ± 3.0 |  | 8.32 ± 0.17 |
| Gaussian, steered to 6 directions | *G* | 0.897 ± 3.2 | 0.903 ± 3.1 | 8.76 ± 0.16 | 8.40 ± 0.16 |
| *H* | 0.820 ± 1.9 | 0.892 ± 2.8 | 16.10 ± 0.43 | 8.85 ± 0.23 |
| *G + H* |  | 0.903 ± 2.9 |  | 8.31 ± 0.16 |
| Gabor, 6 directions | *Re* |  | 0.907 ± 3.1 |  | 7.64 ± 0.13 |
| *Im* |  | 0.895 ± 3.0 |  | 7.87 ± 0.23 |
| *Re + Im* |  | 0.908 ± 3.0 |  | 7.51 ± 0.13 |
| DT-CWT | *Re* |  | 0.903 ± 3.0 |  | 7.80 ± 0.14 |
| *Im* |  | 0.891 ± 3.0 |  | 8.06 ± 0.21 |
| *Re + Im* |  | 0.905 ± 2.9 |  | 7.65 ± 0.13 |

|  |  |  |  |
| --- | --- | --- | --- |
| Fibre data | | | |
|  | | Vessel Detection  *(w = 3)* | Vessel Orientation  *(w = 3)* |
| Gaussian, steered to 6 directions | *Mag* | 0.876 ± 2.5 | 8.65 ± 0.19 |
| *Phase* | 0.876 ± 3.5 | 9.34 ± 0.21 |
| *Mag + Phase* |  | 8.31 ± 0.16 |
| *… using |Im|* | 0.903 ± 3.1 | 8.15 ± 0.17 |
| *… rotated* | 0.900 ± 2.7 | --- |
| *… interpolated* | 0.901 ± 3.1 | 8.57 ± 0.18 |
| Gabor, 6 directions | *Mag* | 0.884 ± 2.5 | 7.23 ± 0.14 |
| *Phase* | 0.867 ± 3.6 | 10.4 ± 0.26 |
| *Mag + Phase* | 0.905 ± 3.1 | 7.07 ± 0.14 |
| *… using |Im|* | 0.908 ± 3.0 | 7.05 ± 0.13 |
| *… rotated* | 0.906 ± 2.6 | --- |
| *… interpolated* | 0.899 ± 2.9 | 7.93 ± 0.17 |
| DT-CWT | *Mag* | 0.869 ± 2.2 | 7.49 ± 0.15 |
| *Phase* | 0.845 ± 4.4 | 10.7 ± 0.21 |
| *Mag + Phase* | 0.898 ± 2.6 | 7.26 ± 0.16 |
| *… using |Im|* | 0.902 ± 2.6 | 7.19 ± 0.15 |
| *… rotated* | 0.899 ± 2.3 | --- |
| Monogenic | | 0.883 ± 2.5 | 12.10 ± 0.35 |
| All filters | | 0.907 ± 2.5 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Fibre data | | | |
|  | | Vessel Detection  *(w = 3)* | Vessel Orientation  *(w = 3)* |
| Gaussian, steered to 6 directions | *1* | 0.851 ± 0.0032 | 12.40 ± 0.38 |
| *2* | 0.885 ± 0.0032 | 9.27 ± 0.23 |
| *4* | 0.893 ± 0.0033 | 8.48 ± 0.18 |
| *8* | 0.839 ± 0.0029 | 11.70 ± 0.17 |
| *16* | 0.706 ± 0.0028 | 19.70 ± 0.34 |
| Gabor, 6 directions | *1* | 0.814 ± 0.0032 | 17.00 ± 0.54 |
| *2* | 0.866 ± 0.0031 | 10.50 ± 0.31 |
| *4* | 0.893 ± 0.0031 | 7.50 ± 0.18 |
| *8* | 0.887 ± 0.0033 | 7.91 ± 0.14 |
| *16* | 0.806 ± 0.0019 | 12.2 ± 0.20 |
| DT-CWT | *1* | 0.781 ± 0.0026 | 18.90 ± 0.58 |
| *2* | 0.847 ± 0.0027 | 10.30 ± 0.27 |
| *4* | 0.874 ± 0.0031 | 7.66 ± 0.18 |
| *8* | 0.860 ± 0.0030 | 8.87 ± 0.15 |
| *16* | 0.763 ± 0.0021 | 13.80 ± 0.25 |

Experiment 3

* DRIVE
  + State of the art detection
* STARE
  + Classify using DRIVE, still good performance
  + Errors for gabor\_a, w = 3, sampling = 0.75, abs mean: 8.83, abs\_median: 4.74
* Fibre
  + Thin and allow tolerance as per Dabbah paper

Discussion

Using estimate of orientation accuracy?

1. We consider transforming the responses to be rotationally invariant for the purposes of segmentation in section X [↑](#footnote-ref-1)