Computer Aided Diagnosis in Breast Histopathology

Jesse Knight, April Khademi
Image Analysis in Medicine Lab, University of Guelph
2015-12-09

Histopathology continues to provide clinically relevant and inexpensive prognostic information for breast cancer tumours. Unfortunately, the inter-oberserver agreement for the Nottingham Grading System (NGS) is low. Digital pathology analysis has the potential to improve scoring repeatability and reduce human bias. To this end, we present a classification algorithm for hematoxylin and eosin stained slides to differentiate between malignant and benign tumours; it achieves 77.6% accuracy on a 58-image database. The method employs only three features, inspired by the NGS criteria, which quantify nuclear structure and texture. We estimate a binary mask of nuclei pixels in the image using an active contours model, after image-specific stain deconvolution, and extract features directly from the nucleuar mask. Images labels are assigned using a Bayesian classifier, which has good generalization performance in leave-one-out validation (74.1% accuracy). We anticipate accuracy improvements with single-nucleus identification from clusters of nuclei, which are currently grouped as single objects in the segmentation. This will facilitate the characterization of more robust structural features, as well as important pleomorphic metrics like circularity and size.

1 Introduction

1.1 Breast Cancer

Breast Cancer is the second leading cause of cancer death in U.S. women; in 2012, an estimated 40,000 deaths were attributed to the disease [1]. However, breast cancer survival rate is highly dependent on the prognosis of tumour(s); while the 5 year survival rate for localized tumours is 98.6%, it is only 23.3% for metastatic cancer [1]. These statistics have motivated large scale investment in early mammographic screening and therapeutic research [1]. The range of prognostic indicators which have been identified and correlated with treatments and outcomes is now broad, facilitating personalized treatment schema [2].

1.2 Histopathology

Histopathology is the analysis of stained, biopsy-extracted tissue samples under a microscope. Despite advances in genomic tumor profiling, histological grading continues to be a reliable and inexpensive indicator for breast cancer prognosis [2, 3]. High grade tumours are typically more aggressive and associated with increased likeliness of metastasis [2]. Tumour grade is determined using semi-quantitative morphological features in hematoxylin and eosin-stained tumor tissue sections [3].

While a variety of grading schemes have been developed, there has been recent pressure to harmonize under a single scheme for consistency across research and clinical fields [3]. The Nottingham Grading System (NGS) has been recommended by many reviewers, having been strongly validated over four decades [2, 3]. The NGS criteria (Table 1) include super-cellular and unicellular structural and texture features, and the tumor grade is defined using the sum of all feature scores for the slide:

- Grade 1: Scores 3 – 5
- Grade 2: Scores 6 – 7
- Grade 3: Scores 8 – 9

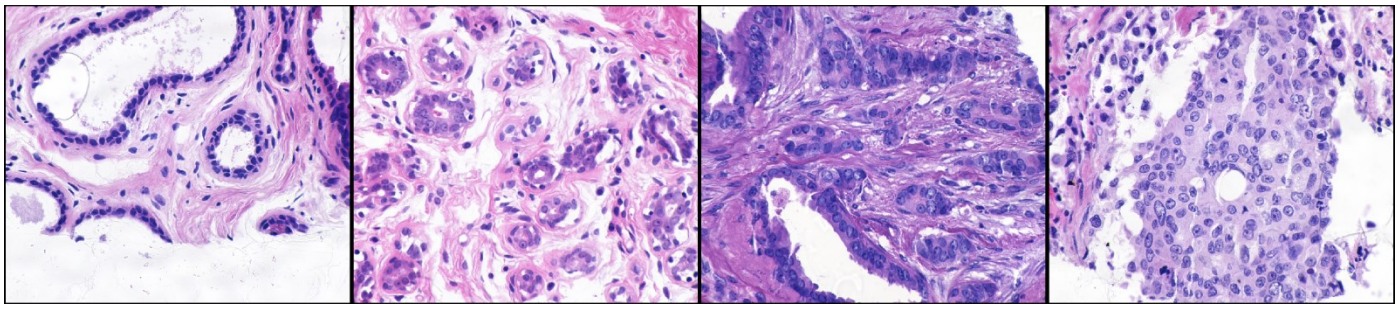

Figure 1: Left to right: increasing histological features of malignancy: diminishing tubule structure, increasing nucleus texture. (left two images: benign, right two images: malignant).

Low grade is associated with benign tumours, while high grade is indicative of malignancy. Some of the readily observable features are shown in Figure 1.

Table 1: Nottingham Grading System semi-quantitative scoring criteria for breast cancer

Feature		Score
Tubule Formation	Majority of tumour (> 75%)	1
	Moderate degree (10 – 75%)	2
	Little or None (< 10%)	3
Nuclear Pleomorphism	Small, regular uniform cells	1
	Moderate increase in size and variability	2
	Marked Variation	3
Mitotic Count	(Microscope field dependent, linear)	1 – 3

Unfortunately, NGS grading is inherently subjective; inter-observer agreement is notoriously low and variable, with kappa statistics ranging from 0.23 – 0.84, potentially leading to suboptimal treatment [3, 4].

This challenge has stimulated research into the development of automated methodologies for histological grading [5]. In this paper, we present such an algorithm, which draws inspiration directly from the NGS feature set (Table 1) to classify pathology images as either malignant or benign.

2 Methods

2.1 Image Database

We develop and test our algorithm on a database of 58 labeled images with 32 benign and 26 malignant cases. Images are sourced from a variety of centres using different stain vendors and protocols, representative of the unfortunate diversity of digitized pathology images.

2.2 Algorithm Overview

We present a simple automated three-feature classifier for differentiating between benign and malignant breast cancer pathology images. Our features are:

- An interstitial space feature
- A nuclear cluster count
- A nucleus texture feature

A high-level abstraction of the processing steps for each image is as follows:

1. Stain hue estimation and deconvolution
2. Nucleus segmenation
3. Feature extraction
4. Classification

2.3 Stain Deconvolution

We begin by computing estimated hematoxylin and eosin stain images using colour deconvolution, as demonstrated in [6, 7]. Due to vendor and preparation variability in stain hues and intensities, this requires a per-image estimate of the stain absorption factors (Figure 2) [7]. For this, we use K-means clustering on the 1 dimensional huespace (histogram) of the image data to define an initial estimate of class membership per pixel. The hue transform is computed,

$$H = \begin{cases} H_1 \\ 2\pi - H_1 \end{cases}, \qquad H_1 = \cos^{-1}\left(\frac{1}{2}\frac{(R-G)(R-B)}{\sqrt{(R-G)^2(R-B)(G-B)}}\right) \tag{2.1}$$

We perform K-means clustering on the hue data, manually seeding three hues $\vec{h} = \{0.61, 0.78, 1.00\}$ (i.e. K = 3) corresponding to typical hematoxylin and eosin hues, as well as a third class: no stain. We use the median RGB values of the pixels in each resulting class k: $\{\bar{R}_k, \bar{G}_k, \bar{B}_k\}$ to define the vector of stain optical densities Λ_k for each stain [6],

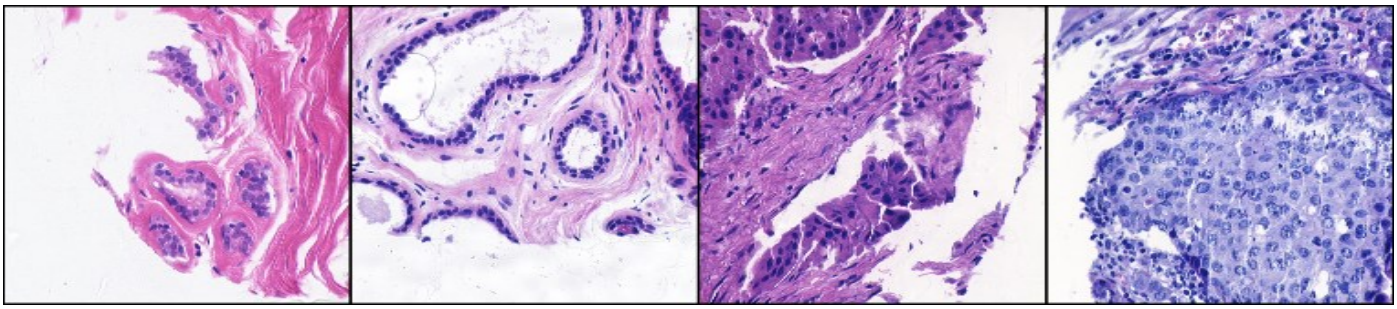

Figure 2: Variability in stain uptake showing necessity of per-image stain optical density estimation for colour deconvolution.

$$\Lambda_k = \begin{bmatrix} d_{R,k} \\ d_{G,k} \\ d_{B,k} \end{bmatrix} = \begin{bmatrix} \bar{R}_k \\ \bar{G}_k \\ \bar{B}_k \end{bmatrix} \tag{2.2}$$

where Λ_3 (no stain) ideally approaches $[1,1,1]^{\mathrm{T}}$. The estimated colour deconvolution matrix $\widehat{D}$ can then be defined as simply the inverse of the concatenated optical density vectors, after normalization,

$$\widehat{D} = \left[\frac{\Lambda_1}{\|\Lambda_1\|}, \frac{\Lambda_2}{\|\Lambda_2\|}, \frac{\Lambda_3}{\|\Lambda_3\|} \right]^{-1} \tag{2.3}$$

Using this matrix, the output image Ψ_{HEX} can be computed in the colourspace with dimensions: hematoxylin, eosin, and no stain – i.e. the three channels correspond to these optical densities. Each colour pixel in this space is computed using the input colour $\Psi_{\mathrm{RGB}}(c)$ and deconvolution matrix $\widehat{D}$,

$$\Psi_{\mathrm{HEX}}(c) = \widehat{D}\, \Psi_{\mathrm{RGB}}(c) \tag{2.4}$$

2.4 Nuclus Segmentation

Cell nuclei are highlighted in the hematoxylin channel due to their affinity for the stain [7]. A large variety of methods have been proposed for segmenting nuclei [5, 8]. However, we employ an active contour model on the hematoxylin image $\widetilde{\Psi}_{\mathrm{H}}$, due to its simplicity and robustness. We first perform normalization on the stain image, comprising of Gaussian low pass filtering, followed by whitening: mean μ subtraction and division by the standard deviation σ,

$$\widetilde{\Psi}_{\mathrm{H}} = \frac{\left(\Psi_{\mathrm{G,H}} - \mu(\Psi_{\mathrm{G,H}})\right)}{\sigma(\Psi_{\mathrm{G,H}})}, \qquad \Psi_{\mathrm{G,H}} = \Psi_{\mathrm{H}} \otimes G \tag{2.5}$$

where $\otimes$ denotes convolution and G is a 3×3 pixel Gaussian kernel with $\sigma = 0.001((n_x + n_Y)/2)$ pixels.

To give an estimate of the binary nucleus segmentation image N', we use the `activecontour` function in MATLAB with the Chan-Vese cost function [9], which has no shrinkage bias. We seed the solution with a blind intensity-based estimate of the image N'' using the normalized hematoxylin image, and iterate to 200 steps or convergence.

$$N'' = \begin{cases} 1, & \widetilde{\Psi}_{\mathrm{H}} > 1.2 \\ 0, & \widetilde{\Psi}_{\mathrm{H}} \leq 1.2 \end{cases} \tag{2.6}$$

$$N' = \texttt{activecontour(}\, \widetilde{\Psi}_{\mathrm{H}}, N'', \texttt{200);} \tag{2.7}$$

Lastly, to clean up the active contours output image N' and give the final nucleus mask N, we perform morphological closing and then opening with a disk kernel K having $r = 2.5$ pixels,

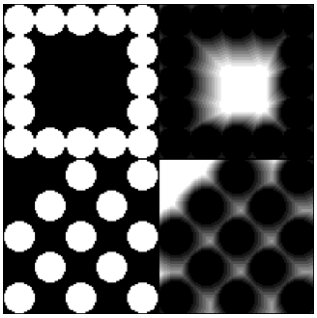

Figure 3: Toy example showing how the interstitial feature quantifies lack of structure. Top: arbitrary structure; bottom: even nuclei distribution. Right images show iterative erosion of the interstitial spaces, where the cumulative number of spaces is counted as the feature.

$$N = \overbrace{\left(\left(\overbrace{\left((N' \oplus K) \ominus K\right)}^{closing} \ominus K\right) \oplus K\right)}^{opening} \tag{2.8}$$

2.5 Feature Extraction

Before including any feature f in the model, we manually validated its correlation with the output target (malignancy label) $Y \in \{0,1\}$ for all images using Pearson's correlation coefficient. Features failing to reach the significance level of $\alpha = 0.05$ were omitted. This hand-designed approach is contrasted with machine learning approaches, which rely on automatic dimensionality reduction to resolve relevant manifolds in many-feature spaces [5].

2.5.1 Interstitial Space Feature (ISS)

We aimed to design a rotation, scale, and shift invariant feature to quantify the degree of structure in the arrangement of nuclei, emulating the tubule formation criteria in the NGS. To this end, we have constructed a novel morphological metric which is relatively robust.

The measure is a count of interstitial spaces between nuclei in the segmentation image. The count is repeated each time after dilating the image N, and summed over all iterations. Figure 3 demonstrates how this penalizes large contiguous spaces (e.g. tubules) while yielding large values for evenly distributed (unstructured) nuclei.

```
for k = 1:steps
    Nimg = imdilate( Nimg, K );
    [ ~ , inter(k) ] = bwlabel( 1 - Nimg );
end
feat = sum( inter(:) );
```

We dilate with a disk kernel K having radius $r = 2$ pixels, and iterate for 50 steps. The MATLAB function `bwlabel` is used to count the interstitial spaces.

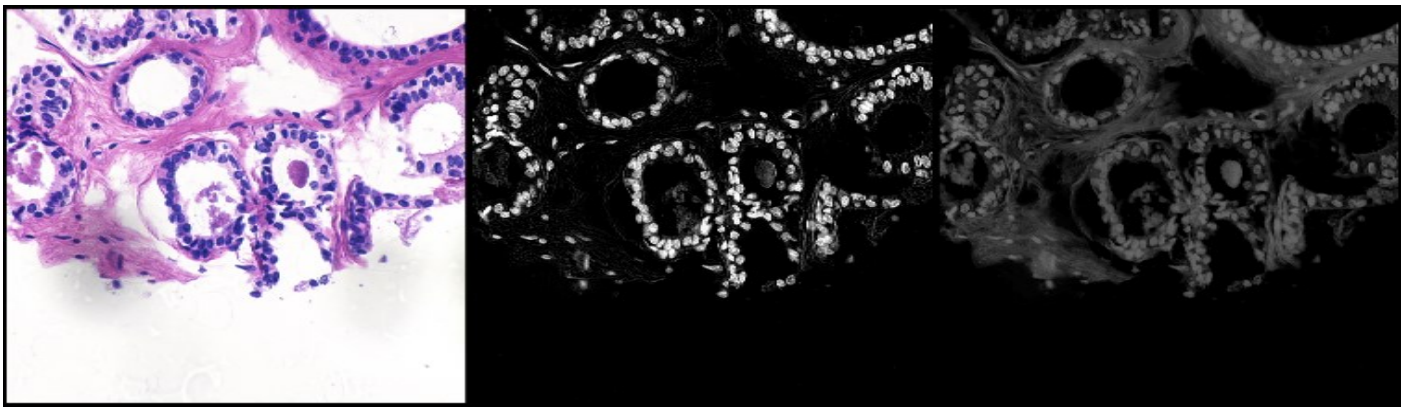

Figure 4: Results of colour deconvolution: hematoxylin and eosin stain density images (third *no stain* image is omitted).

2.5.2 Nucleic Object Count (NOC)

We also compute a very simple secondary structural metric: the total number of unique binary objects in the image N. Due to the morphological closing used during nucleus segmentation, super-cellular structures will tend to have contiguous masks. Therefore, as above, a high number of objects would be correlated with malignancy.

2.5.3 Nucleus Texture Feature (TEX)

To quantify nuclear texture, we consider only pixels in the segmentation mask N. However the same characteristic of the nucleus mask N which facilitates the second feature – morphological connectivity – will introduce artificial texture due to overlapping nuclei. We therefore exclude all segmentation objects with mass too large or exceedingly small with the aim of considering only isolated nuclei. We use 500 and 50 pixel thresholds, respectively. We also erode the map by 2 pixels to remove non-overlapping edges.

We then compute the texture metric, reflective of the pleomorphic criteria in the NGS, within the conservative nucleus mask. We filter the HSV image value channel Ψ_V with a texture kernel R, and average the absolute value of the filtered image over the whole mask. The kernel R is the 2D ripple texture kernel, as defined by Law in 1980 [10],

$$\mathrm{R} = \mathrm{R5}^{\mathrm{T}}\mathrm{R5}, \qquad \mathrm{R5} = [+1, -4, +6, -4, +1] \tag{2.9}$$

$$f = |(\Psi_V \otimes R) \cdot N_e|_1 \tag{2.10}$$

2.5.4 Omission of Mitotic Feature

We have not attempted to quantify mitosis, as automated mitotic cell identification is an extremely challenging problem currently addressed only by state-of-the-art neural network approaches [11]. Furthermore, the small size of the images in our database means that they may not contain statistically useful numbers of mitotic events.

2.6 Classification

Finally, we compute the feature representation of all images, and construct a machine learning classifier to assign a malignancy label per image. We use a Naïve Bayesian classifier, as other nonlinear classifiers (e.g. Random Forest) tended to over fit the small training set.

2.7 Performance Analysis

In the event of any processing step failure, we did not perform any manual correction in order to demonstrate the performance of the fully automated algorithm.

2.7.1 Nucleus Segmentation

The nucleus segmentation N is evidently fundamental to all of our features; we therefore performed basic validation on a single mask. We selected an image with a large number of polymorphically typical nuclei (texture, structure, contrast) for manual nucleus delineation. We then computed the Dice similarity coefficient (DSC), sensitivity and specificity for the estimated segmentation on the same image.

$$DSC = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{2.11}$$

$$Sen = \frac{TP}{TP + FN}, \qquad Spe = \frac{TN}{TN + FP} \tag{2.12}$$

2.7.2 Malignancy Classification

The goal of our algorithm is to classify images of malignant histopathology. We computed the accuracy, sensitivity, and specificity of the malignancy label over all image examples,

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.13)$$

Additionally, to demonstrate that the model did not overfit our small training database, we performed leave-one-out validation. We removed each image, one at a time, from the feature space data used to construct the Bayesian classifier, and then used the classifier to label the missing image.

3 Results

3.1 Stain Deconvolution

Stain images for hematoxylin and eosin, resulting from colour deconvolution are shown in Figure 4. The image-specific stain hue estimation failed drastically in only 3 / 58 (5%) of images; in these instances, the hematoxylin image had inverse contrast (dark nuclei, bright background), making the failure obvious. Evidently, these 3 images would have little chance of correct nucleus segmentation and subsequent feature extraction. We did not explicitly validate the performance of the eosin or no stain channels, as these were not relevant to the remainder of the processing steps.

3.2 Nucleus Segmentation

Based on our single ground truth image comparison (which did not fail colour deconvolution), our method for nucleus segmentation is highly specific (99%) but not sensitive (77%) and achieves a DSC of 0.84. A zoom of some structured and distributed nuclei in the comparison is shown in Figure 5.

3.3 Feature Set

We aimed to design a set of scale, rotation and shift invariant features which are mutually orthogonal and also highly correlated with the malignant label. Our features are all shift and rotational invariant, with the exception of slight differences in the texture image every 45° due to the square kernel R. Moreover, our features can all be made scale invariant with appropriate normalization.

The correlation statistics between features and with the output label are presented in Table 2. We also performed singular value decomposition on the feature space of to quantify the relative variance in orthogonal directions. The resulting matrix of singular values Σ had diagonal values $\{1.70, 0.30, 0.14\}$, yielding a minimum ratio of variances of 0.14 / 1.70 = 0.08. This indicates that the minimally variant direction in the feature space is still has 8% of the variance found in the principal direction, implying further dimensionality reduction may lose some significant information in the data.

Table 2: Correlation statistics of feature set

Feature	Mutual Correlation			Malignancy Correlation
	ISS	NOC	TEX	
ISS	–	0.825	0.101	0.496 ($p < 0.001$)
NOC	0.825	–	0.082	0.452 ($p < 0.001$)
TEX	0.101	0.082	–	0.275 ($p = 0.036$)

3.4 Classification Performance

Our three-feature approach achieves a respectable accuracy of 77.6%. The algorithm has roughly equal error rates for malignant images and benign images, correctly classifying 20 / 26 malignant images and 25 / 32 benign images (Table 3). Leave-one-out validation resulted in only a 3.5% error increase, attributable to an increase in false negatives – i.e. decreased sensitivity, no change in specificity.

Table 3: Confusion matrix and performance statistics for image malignancy classification with: (Full) full dataset to train and test and (LOO) leave-one-out validation.

			Predicted		Performance		
			+	–	Acc	Sen	Spe
Full	Truth	+	20 (34%)	6 (10%)	77.6%	76.9%	78.1%
		–	7 (12%)	25 (43%)			
LOO	Truth	+	18 (31%)	8 (14%)	74.1%	69.2%	78.1%
		–	7 (12%)	25 (43%)			

The feature space and Bayesian decision surface are depicted in Figure 6, showing the distribution, separability and estimated classification of images based on our features.

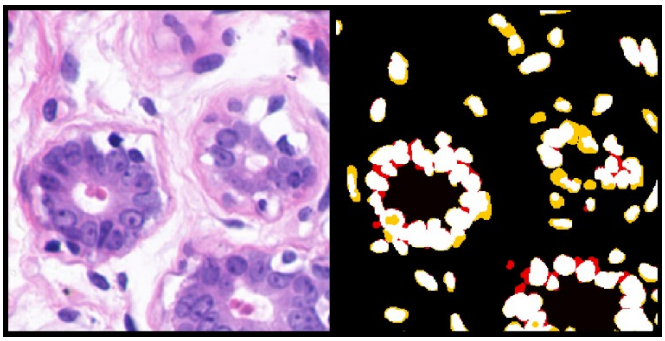

Figure 5: Nucleus segmentation validation showing (white) agreement with ground truth, (red) overestimation and (yellow) underestimation.

4 Discussion

4.1 Nucleus Segmentation

That our nucleus segmentation approach is more specific than sensitive is desirable. Given ensuing texture and structural features, it is more important that false positives remain low than we mistakenly identify false nuclei. From, Figure 5 we also observe that the lack of sensitivity typically results from erosion of edges (only one nucleus is missed altogether). This is also desirable, as the spatial distribution of the nuclei is captured – relevant to structural features – if not their shape and size – which is not used in our current method. This small amount of erosion, in fact, simply represents a negative shift along the dilation iteration dimension during interstitial space counting (Figure 7 (C)).

4.2 Interstitial Space Feature

The novel morphological structure feature proposed here is happily shift, rotation and scale invariant. However, from Figure 7 (C) we observe that the sum over the iteration interstitial space counts actually lowers the contrast between structured and unstructured tissues. Rather, large spaces, representing structured nuclei, should be counted separately from small spaces, which are indicative of unstructured cells. This could be easily achieved using a size, threshold, or equivalently, a threshold along the dilation step dimension, as only large spaces persist past many dilation steps. Importantly, there is no reason to assume that the two resulting features would be correlated.

4.3 Decision Tools & Validation

The development of a classifier with true generalization performance is not to be taken for granted. The data set here, while diverse, is far too small for training a truly robust classifier. For instance, we constructed a 10-tree random forest decision tool using the `TreeBagger` object class in MATLAB, and labeled the data using the three feature model. The performance was 98% accuracy, as this type of classifier has a high capacity to overfit training data (this was explicitly omitted from the Results section as it is not a valid result). Even the Naïve Bayes classifier we employed was likely unduly biased by the malignant outlier at the top of the feature space in Figure 6, which pulled the decision surface strongly towards it.

Furthermore, it was noted that this image set is one of the standard testing databases for breast cancer digital pathology. This motivates making a strong distinction between training and testing databases, and developing large standardized image sets for each, as has been achieved in the machine learning community. The notion of proprietary images and exclusive results which currently prevents this approach is toxic to the development of high quality research and should not be tolerated.

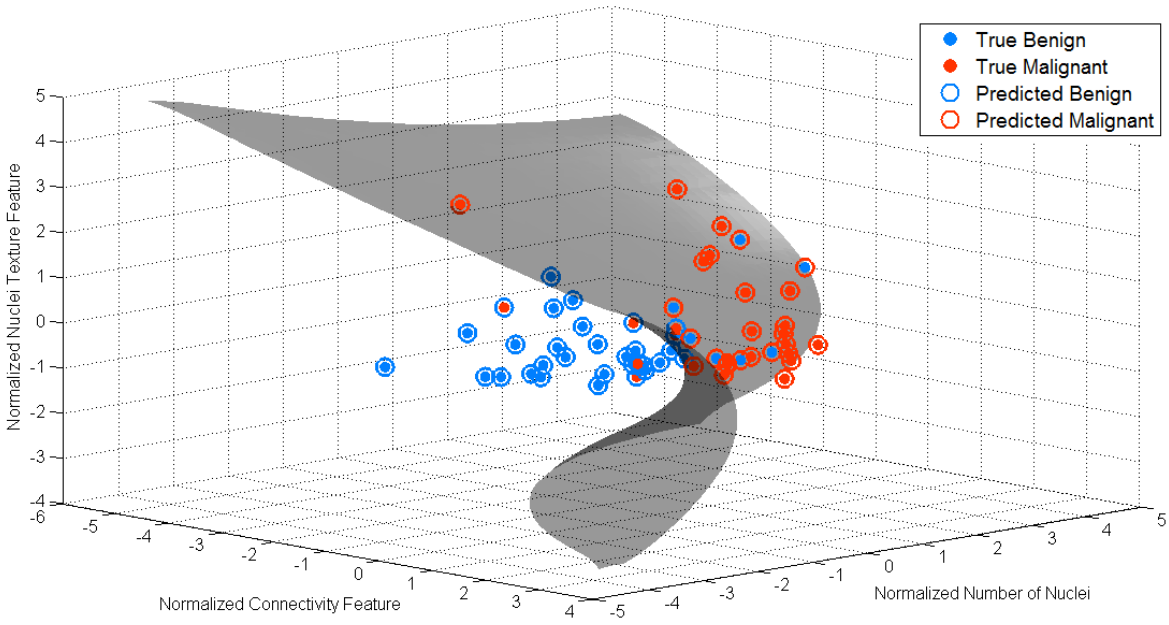

Figure 6: Bayesian classifier decision surface showing ground truth labels and predicted labels for the 58 image database

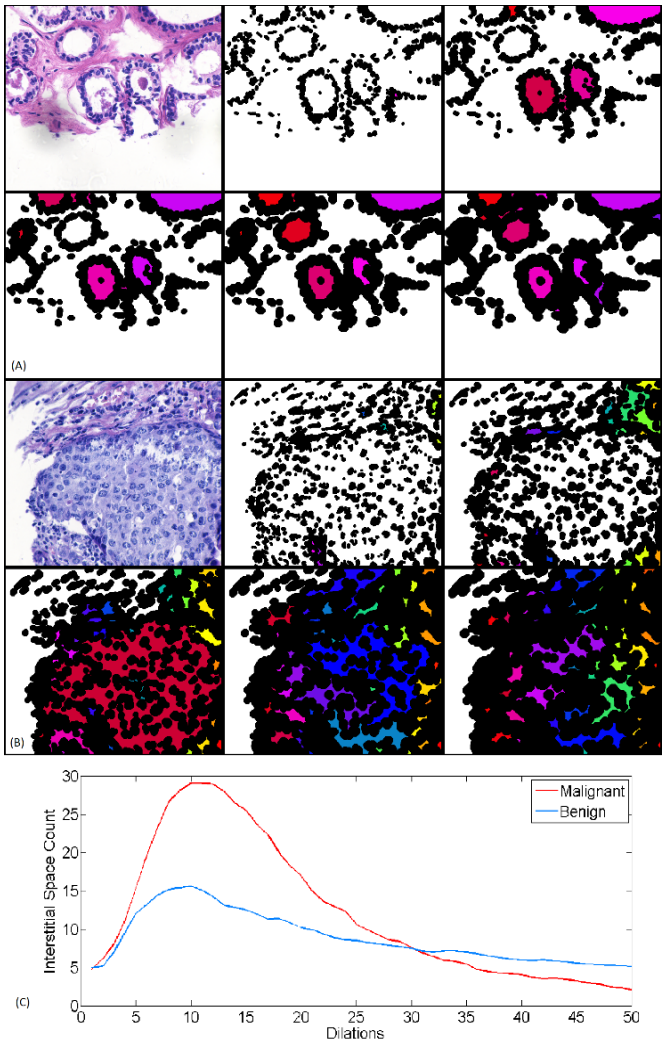

Figure 7: Interstitial space count feature; progression of nucleus segmentation mask dilation in (A) benign and (B) malignant images – rainbow colour is proportional to object count; (C) average counts over dilation iterations.

5 Future Works

Whereas histopathology assessments are often ordered to give a definitive diagnosis or prognosis for a patient, the accuracy of our algorithm is much too low to be of clinical use. We now present some obvious avenues of potential improvement.

5.1 Stain Estimation

Our per-image stain estimation using K-means clustering failed drastically on 3/58 images (5%). This prevents subsequent processing steps from extracting any useful features, and represent a significant challenge for the model currently. We recommend stain colour normalization, as has been proposed in [7] for more robust deconvolution.

5.2 Feature Refinement

5.2.1 Structural Features

As noted above, the novel morphological interstitial space feature holds promise, but should be separated into two distinct features using a size threshold. This would likely produce two uncorrelated features, facilitating separation of classes with more certainty in a higher dimensional space.

Similarly, the nucleic object count feature could benefit from normalization to the total area of the object mask, so as not to bias images with many nuclei towards malignancy. However, increased number of nuclei due to proliferation may be actually associated with aggressive pathologies, so potentially normalization should not be employed.

5.2.2 Nuclear Pleomorphism Features

Our pleomorphism feature does not consider nuclear shape or size, features specifically outlined in the NGS scheme. A method to recognise individual nuclei would therefore be of great benefit, as it would circumvent the challenges of grouped nuclei in our current method, which prevent shape and size analysis.

Additionally, while we considered lower order statistical moments, entropy, Gabor filters, and Law's texture features during model development, a wide variety of other texture metrics are available, including local binary patterns and wavelet decomposition. Noting that the correlation between our texture feature and the malignant label was only marginal ($p = 0.036$), further investigation should consider alternative texture measures.

5.3 Training Data

As noted above, while diverse, our image dataset is extremely small. Increasing the number of examples (i.e. more densely filling the feature space in Figure 6) would enable a more nuanced understanding of the performance of each feature, and facilitate automatic optimization of model parameters. Moreover, classifier generalization performance is proportional to the training set size. Classifiers with higher capacity could also be considered (e.g. random forests) – which could not be employed here due to overfitting.

6 Conclusion

We have demonstrated the feasibility of a computer aided diagnosis tool for classification of digitized breast cancer histology slides. Our algorithm draws inspiration from the widely used Nottingham Grading Scale to define two structural features and one nuclear texture feature, including a novel morphological feature which is scale, shift, and rotational invariant.

We use image-specific colour deconvolution to estimate the hematoxylin stain image, compute a nuclei segmentation mask form this image, and then perform feature extraction. All three features are correlated with malignancy ($p < 0.05$), and have moderate mutual orthogonality.

We achieve an accuracy of 77.6% on the full dataset and 74.1% using leave-one-out validation. Our model is nearly equally sensitive and specific when identifying malignant images, indicating that it has minimal bias towards either. While this accuracy is not yet high enough for clinical use, a number of potential algorithmic improvements have been proposed, the development of which is may lift the model into utility.

Lastly, we strongly advocate for the development of larger, standardized image databases for training and testing digital pathology analysis methods, emulative of the machine learning community. Small image sets strongly encourage overfitting using many statistical classification tools, yielding inflated results which are likely to be published.

7 References

[1] J. Ma and A. Jemal, "Breast Cancer Statistics," in *Breast Cancer Metastasis and Drug Resistance*, New York, Springer Science, 2013, pp. 1-18.

[2] C. W. Elston and I. O. Ellis, "Pathological prognostic factors in breast cancer. I. The value of histological grade in breast cancer: experience from a large study with long-term follow up," *Histopathology,* vol. 13, pp. 403-410, 1991.

[3] E. A. Rakha, J. S. Reis-Filho, F. Baehner, D. J. Dabbs, T. Decker, V. Eusebi, S. B. Fox, S. Ichihara, J. Jacquemier, S. R. Lakhani, J. Palacios, A. L. Richardson, S. J. Schnitt, F. C. Schmitt, P.-H. Tan, G. M. Tse, S. Badve and I. O. Ellis, "Breast cancer prognostic classification in the molecular era: the role of histological grade," *Breast Cancer Research,* vol. 12, no. 207, 2010.

[4] A.-M. O'Shea, E. A. Rakha, A. Hodi, I. O. Ellis and A. H. S. Lee, "Histological grade of invasive carcinoma of the breast assessed on needle core biopsy - modifications to mitotic count assessment to improve agreement with surgical specimens," *Histopathology,* vol. 59, pp. 543-548, 2011.

[5] S. Doyle, S. Agner, M. Feldman, J. Tomaszewski and M. Anant, "Automated grading of breast cancer histopathology using spectral clustering with textural and architextural image features," *ISBI,* pp. 496-499, 2008.

[6] A. C. Ruifrok and D. A. Johnston, "Quantification of histochemical staining by color deconvolution," *Anal Quant Cytol Histol* , vol. 23, no. 4, pp. 291-299, 2001.

[7] A. M. Khan, N. Rajpoot, D. Treanor and D. Magee, "A Non-Linear Mapping Approach to Stain Normalizatoin in Digital Histopathology Images using Image-Specific Colour Deconvolution," *TBME, IEEE,* 2014.

[8] M. Yadollahi and A. Prochazka, "Image segmentation for object detection," *MATLAB Conference Proceedings* , 2011.

[9] T. F. Chan and L. A. Vese, "Active contours without edges," *Image Processing, IEEE Transactions on,* vol. 10, no. 2, pp. 266-277, 2001.

[10] K. I. Laws, "Textured Image Segmentation," University of Southern California Los Angeles, Image Processing Inst., Los Angeles, 1980.

[11] S. Huh, D. F. Elmer Ker, R. Bise, M. Chen and T. Kanade, "Automated Mitosis Detection of Stem Cell Populations in Phase-Contrast Microscopy Images," *Medical Imaging, IEEE Transactions on,* vol. 30, no. 3, pp. 586-596, 2011.

8 Appendix: MATLAB Code

8.1 (RTF) Usage Manual

8.1.1 Pipeline

The user should only be required to run `[P, F, Y, NB, Yp] = pipeline;` once. This will create 4 checkpoint `.mat` files, which contain the image data at various stages of processing (i.e. with redundant duplications between `.mat` files). The files total 1.3 GB of memory. Once the pipeline is complete, loading the last file (`P_cdc_nseg_feat.mat`) will load all the information from the processing pipeline (`P, F, Y, NB, Yp`) – i.e. the other files can be deleted.

8.1.2 Data Structures

- `P` is an array of all intermediary processing step images for all source images – e.g. to access the image `nuc` (hematoxylin channel) for the 15th image, the syntax is `P(15).nuc`. The fields of `P`, after all processing steps are complete, are given in Table 4. Note that this single structure requires 840 MB in MATLAB.
- `Y` is the vector of true binary malignancy labels.
- `F` is a matrix of features with dimensions (58 × 3); the first dimension corresponds to the images (as in array `P`), and the second to the feature number, in the order: $\{f_{ISS}, f_{NOC}, f_{TEX}\}$.
- `NB` is a Naïve Bayes classifier object (MATLAB).
- `Yp` is the vector of predicted binary malignancy labels.

Table 4: Image structure P fields

Field	Symbol	Description	Field	Symbol	Description
`.img`	Ψ_{RGB}	Original RGB MxNx3 image	`.nucseg`	N	Binary mask of segmented nuclei
`.mal`	Y	Binary malignancy label	`.f_iss`	f_{ISS}	Vector of 50 interstitial space counts
`.fname`	–	Original image filename	`.f_noc`	f_{NOC}	Count of nuclear objects
`.nuc`	Ψ_H	Grayscale MxN hematoxylin stain image	`.f_tex`	f_{TEX}	Texture feature magnitude
`.cyt`	Ψ_E	Grayscale MxN eosin stain image			

8.1.3 Output

`timshow` is a flexible function useful for visualizing many outputs simultaneously – e.g. `timshow(P(1:16).nuc, hot, [0,10])` will show the hematoxylin channel for the first 16 images with the `hot` colourmap, contrast scaled to the range `[0,10]`. Similarly, `timshow(P((Yp-Y) == +1).img)` will show all false positive images, and `timshow(P((Yp-Y) == -1).img)`, false negatives. The function also accepts many image arguments explicitly, so `timshow(P(1).img, P(1).nuc, P(1).cyt)` is fine.

Classification performance can be visualized using `validateclassify`, which includes generating the 3D rendering show in Figure 6. Nucleus segmentation can be examined using `validatenucseg` (the manually segmented image for `P(1)` has been provided in the folder `/manualnucseg/`, so the user must pass this function `P(1)`).

8.2 Table of Contents

This digital table is hyperlinked for convenience.

8.3 Processing Pipeline

```
function [P, F, Y, NB, Yp] = pipeline()
imgdir  = 'img/';   % Path to src img dir
datadir = 'data/';  % .mat data save dir
                    % (use checkpoints + comments to skip steps after first run)
checkpoint_raw  = [datadir,'P.mat'];
checkpoint_cdc  = [datadir,'P_cdc.mat'];
checkpoint_nseg = [datadir,'P_cdc_nseg.mat'];
checkpoint_feat = [datadir,'P_cdc_nseg_feat.mat'];

if ~exist(datadir,'dir')
    mkdir(datadir);
end

% load all images (P(k).img) and labels (P(k).mal, Y)
if exist(checkpoint_raw,'file')
    if ~exist(checkpoint_cdc,  'file') && ...
       ~exist(checkpoint_nseg, 'file') && ...
       ~exist(checkpoint_feat, 'file')
        fprintf('...LOADING DATABASE\n');
        load(checkpoint_raw);
    end
else
    [P,Y] = buildimgdatabase(imgdir);
    fprintf('...SAVING DATABASE\n');
    save(checkpoint_raw,'P','Y','-v7.3');
end

% perform color deconv for all images (P(k).nuc = hem, P(k).cyt = eosin)
if exist(checkpoint_cdc,'file')
    if ~exist(checkpoint_nseg, 'file') && ...
       ~exist(checkpoint_feat, 'file')
        fprintf('...LOADING DATABASE + COLOUR DECONV\n');
        load(checkpoint_cdc);
    end
else
    [P] = batch_clrdeconv(P);
    fprintf('...SAVING DATABASE + COLOUR DECONV\n');
    save(checkpoint_cdc,'P','Y','-v7.3');
end

% perform nucleus segmentation for all images (P(k).nucseg)
if exist(checkpoint_nseg,'file')
    if ~exist(checkpoint_feat, 'file')
        fprintf('...LOADING DATABASE + COLOUR DECONV + SEGMENTATION\n');
        load(checkpoint_nseg);
    end
else
    [P] = batch_nucseg(P);
    fprintf('...SAVING DATABASE + COLOUR DECONV + SEGMENTATION\n');
    save(checkpoint_nseg,'P','Y','-v7.3');
end

% perform feature extraction for all images (P(k)
if exist(checkpoint_feat,'file')
    fprintf('...LOADING DATABASE + COLOUR DECONV + SEGMENTATION + FEATURES\n');
    load(checkpoint_feat);
else
    [P,F] = batch_findfeats(P);
    [NB,
    fprintf('...SAVING DATABASE + COLOUR DECONV + SEGMENTATION + FEATURES\n');
```

```
    save(checkpoint_feat,'P','Y','F','Yp','-v7.3');
end


function [P] = batch_clrdeconv(P)
fprintf('...ASSIGNING COLOR DECONVOLUTION\n');

hueseeds = [0.61; 0.78; 1.00]; % [0.61; 0.78; 1.00]

for p = 1:numel(P)
    fprintf(['P: ',num2str(p),'\n']);

    ch = single( clrdeconv(P(p).img, hueseeds) );
    P(p).nuc = ch(:,:,2);
    P(p).cyt = ch(:,:,3);
end

function [P] = batch_nucseg(P)
fprintf('...ASSIGNING NUCLEUS SEGMENTATION\n');

for p = 1:numel(P)
    fprintf(['P: ',num2str(p),'\n']);

    P(p).nucseg = uint8( nucseg(P(p).nuc) );
end

function [P,F] = batch_findfeats(P)
fprintf('...DEFINING FEATURES\n');

for p = 1:numel(P)
    fprintf(['P: ',num2str(p),'\n']);

    [P(p).f_iss] = feat_iss(P(p).nucseg);
    [P(p).f_noc] = feat_noc(P(p).nucseg);
    [P(p).f_tex] = feat_tex(P(p).img, P(p).nucseg);

    F(p,1) = mean(P(p).f_iss);
    F(p,2) = P(p).f_noc;
    F(p,3) = P(p).f_tex;
end
```

8.4 Loading Images

```
function [P,Y] = buildimgdatabase(dirname)

clrspace  = 'rgb';
imgtype   = '.tif';
labels    = {'benign','malignant'};

P = [];
Y = [];

D = rdir(dirname);

fprintf('...BUILDING DATABASE\n');

for d = 1:numel(D)
    fname = D(d).name;
    if strcmp(filetype(fname), imgtype)
```

```
        fprintf([fname,'\n']);

        img = double(imread(D(d).pathname))./255;
        switch clrspace
            case 'rgb'
                img = img;
            case 'hsv'
                img = rgb2hsv(img);
            case 'lab'
                cxform = makecform('srgb2lab');
                img = applycform(img,cxform);
        end

        P(end+1).img = single(img);
        P(end).mal   = any(strfind(fname,labels{2}));
        P(end).fname = fname;
        Y(end+1)     = P(end).mal;
    end
end
Y = Y';
```

8.5 Colour Deconvolution

```
function [ichan] = clrdeconv(img, seeds)
% estimating the per-image stain optical density vectors
[ipre, ihsv] = preprocessing (img);
[iclass]     = huecluster    (ihsv, seeds);
[icbin]      = class2bin     (iclass);
[classclr]   = classmeanclr  (ipre, icbin);
% deconvolving the image channels
[ichan]      = channelsep    (img, classclr);

function [ipre,ishv] = preprocessing(img)
% basic vector mean filtering
ipre = cdotfilter(img, ones(3), 1, @mean);
ishv = rgb2hsv(ipre);

function [iclass] = huecluster(img, seeds)
% K-means K = 3 cluster the image hue histogram

isize = size(img(:,:,1));
X = reshape(img(:,:,1), [prod(isize),1]);

nfails = 0;
while nfails < 10
    try
        [vclass] = kmeans(X, size(seeds,1), ...
            'distance', 'sqeuclidean', 'start', seeds);
        break
    catch
        nfails = nfails + 1;
    end
end
iclass = reshape(vclass,isize);

function [icbin] = class2bin(iclass)
% explode the class image into binary masks
M = bindisk(2);
C = unique(iclass);
icbin = zeros([size(iclass), numel(C)]);
```

```
for c = 1:numel(C)
    cbin = zeros(size(iclass));
    cbin(iclass == C(c)) = 1;
    cbin = imclose(imopen(cbin,M),M);
    icbin(:,:,c) = cbin;
end

function [classclr] = classmeanclr(img, icbin)
% find medial colours per class (rgb space)
classclr = zeros(size(icbin,3),3);
for i = 1:3
    imchan = img(:,:,i);
    for c = 1:size(icbin,3)
        classclr(c,i) = median(imchan(icbin(:,:,c)==1));
    end
end

function [ichan] = channelsep(img, classclr)
% seperate the channels
nc       = size(classclr,1);
ioptdens = -log(img);
ichan    = zeros(size(img,1),size(img,2),nc);
% optical density matrix
for c = 1:nc
    M(c,:) = classclr(c,:)./norm(classclr(c,:));
end
% inverse transform
D  = pinv(M);
% assigning output image by colour+class (much faster than by x+y)
for c = 1:nc
    for i = 1:size(classclr,2)
        ichan(:,:,c) = ichan(:,:,c) + ioptdens(:,:,i).*D(c,i);
    end
end
```

8.6 Nucleus Segmentation

```
function [segmap] = nucseg(nuc)
% pre-processing: smoothing, whitening
sd     = 0.001 * mean(size(nuc));
nucg   = imfilter(nuc, fspecial('gaussian', round(3*sd), sd));
stdnuc = (nucg - mean(nucg(:))) ./ (std(nucg(:)));
% active contours
acseg = activecontour(stdnuc, stdnuc > 1.2, 200);
% morpholical clean-up
M = bindisk(2.5);
segmap = uint8( imopen(imclose(acseg,M),M) );
```

8.7 Feature Extraction

8.7.1 Interstitial Space Feature

```
function [feat] = feat_iss(nucseg)
% parameters
M      = bindisk(2);
steps  = 50;
thr    = 0.0001 * numel(nucseg);
% initializations
nucseg = single(nucseg);
feat   = zeros(steps,1);
```

```
% iterating
for k = 1:steps
    % dilating and labeling interstitial spaces
    nucseg  = imdilate(nucseg, M);
    [BWL,B] = bwlabel( 1 - nucseg );

    % counting interstitial spaces (ignoring small)
    N = 0;
    for b = 1:B
        idx = (BWL == b);
        tubesize = sum(sum(idx));
        if tubesize > thr
            N = N + 1;
        end
    end
    feat(k) = N;
end
```

8.7.2 Nuclear Object Count Feature

```
function [feat] = feat_noc(nucseg)
% yes it's really that simple...
[~,feat] = bwlabel(nucseg);
```

8.7.3 Nuclear Texture Feature

```
function [feat] = feat_tex(img, nucseg)

hsvimg = rgb2hsv(img);  % hsv image
vimg   = hsvimg(:,:,3); % value image

k1 = [ 1,-4, 6,-4, 1];  % 1D kernel from Laws PhD
K  = k1'*k1;            % making 2D

teximg = abs(imfilter(vimg,K,'same')); % texture magnitude image (Laws' kernel)
nucteximg   = zeros(size(teximg));     % texture image to be masked by nucseg

% refine the nucleus segmentation to be more conservative
nucseg = imerode(nucseg, bindisk(1));  % erode by 2 pix to remove edge pixels*
[objimg,N] = bwlabel(nucseg);          % label objects
for n = 1:N
    idx = (objimg == n);
    omass = sum(idx(:));
    if omass > 50 && omass < 500       % keep only objects with 50 < mass < 500
        nucteximg = nucteximg + (idx.*teximg);
    end
end

% feature is mean of the masked part of the texture image
feat = mean(nucteximg(nucteximg > 0));

% *bindisk(1) has r = 1 --> d = 2
```

8.8 Dependencies

8.8.1 Semantic Functions

```
function [fulldir] = rdir(dirname)
basedir = dir(dirname);
fulldir = basedir(3:end);
```

```
for d = 3:numel(basedir)
    fulldir(d-2).pathname = [dirname,'\',fulldir(d-2).name];
    if exist( [dirname, '\', basedir(d).name], 'dir')
        appendingdir = rdir( [dirname, '\', basedir(d).name] );
        if ~isempty(appendingdir)
            fulldir = [fulldir; appendingdir];
        end
    end
end

function [ext] = filetype(filename)
idx = find((filename == '.'),1,'last');
ext = filename(idx:end);
```

8.8.2 Simple Operations

```
function [B] = bindisk(r)
N = 2*r;
B = zeros(round(2*r));
for y = 1:N
    for x = 1:N
        H = sqrt((y-((N+1)/2))^2 + (x-((N+1)/2))^2);
        if H <= r
            B(y,x) = 1;
        end
    end
end
```

8.8.3 Vector Filtering

```
function [IO] = cdotfilter(I,K,vidx,op)
% explode the pixels specified by the kernel (K) in I into a new dimension
IK = kernelcat(I,K);
% consider all pairs of vectors in K
pair = nchoosek(1:sum(K(:)),2);
% for each vector in K, sum the dot products with every other vector in K
% this is the sorting feature - measure of 'centralness' of the vector
Kdot = zeros(size(IK,1),size(IK,2),size(IK,3));
for p = 1:size(pair,1)
    vdot = dot( squeeze(IK(pair(p,1),:,:,:)), squeeze(IK(pair(p,2),:,:,:)), 3 );
    Kdot(pair(p,1),:,:) = Kdot(pair(p,1),:,:) + shiftdim(vdot,-1);
    Kdot(pair(p,2),:,:) = Kdot(pair(p,2),:,:) + shiftdim(vdot,-1);
end
% sort the feature, store the sorting indexes
[~,sidx] = sort(Kdot,1);
% using the sorted feature, select the sorted vectors specified by user's vidx,
% and perform the user-specified operation op on these vectors to compute the
% output image OUT.
IO = spvop(sidx,vidx,IK,op);

% explode the pixels specified by K in I into a new dimension (prepended)
% KI has dimensions [numel(K nonzero), size(I)]
function [KI] = kernelcat(I, K)
I = double(I);
isize = size(I);
ksize = sum(abs(sign(K(:))));
KI = zeros([ksize,isize]);
C  = round([size(K,1),size(K,2),size(K,3)]/2);

k = 1;
for z = 1:size(K,3)
    for x = 1:size(K,2)
        for y = 1:size(K,1)
```

```
            if(K(y,x,z))
                KI(k,:,:,:) = imshift(I,(C - [y,x,z]));
                k = k + 1;
            end
        end
    end
end

% custom image shifting function because MATLAB's is overcomplicated
function [IS] = imshift(I,T)
pos = sign(T) > 0;
IS = I;
if pos(1) % down
    IS = padarray(IS(1+T(1):end,:,:),[+T(1),0,0],'replicate','post');
else      % up
    IS = padarray(IS(1:end+T(1),:,:),[-T(1),0,0],'replicate','pre');
end
if pos(2) % right
    IS = padarray(IS(:,1+T(2):end,:),[0,+T(2),0],'replicate','post');
else      % left
    IS = padarray(IS(:,1:end+T(2),:),[0,-T(2),0],'replicate','pre');
end
if pos(3) % forward
    IS = padarray(IS(:,:,1+T(3):end),[0,0,+T(3)],'replicate','post');
else      % backward
    IS = padarray(IS(:,:,1:end+T(3)),[0,0,-T(3)],'replicate','pre');
end

function [OUT] = spvop(sidx,vidx,IK,op)
% preallocate the output image
% and exploded output image before OP is performed along the extra dimension
OUT  = zeros(size(IK,2),size(IK,3),size(IK,4));
OUTv = zeros([size(OUT),numel(vidx)]);

% for all desired indexes in the sorted vector list...
% (perform all operations simultaneously across the image pixels)
for v = 1:numel(vidx)
    % consider the vector specified by index vidx(v) in the sorted vectors
    Xv = squeeze(sidx(vidx(v),:,:));
    % sparsely explode the indexes corresponding to the sorted vector
    % into the kernel dimension, and pad by colour dimension
    spidx = padarray(spunique(Xv),[0,0,0,size(IK,4)-1],'replicate','post');
    % the component of the output image contributed by the sorted vector at vidx
    OUTv(:,:,:,v) = sum(spidx.*IK,1);
end
% perform the user-specified operation along the extra dimension of the exploded
% output image to yield the final output image.
OUT = op(OUTv,4);

% encode unique elements along new sparse dimension
function [sp] = spunique(X)
U = unique(X);
sp = zeros([numel(U),size(X)]);
for u = 1:numel(U)
    sp(u,:) = X(:) == u;
end
```

8.9 Augmented Image Display

```
% Jesse Knight 2015
% Copyright: Khademi Lab (Image Analysis in Medicine Lab).
%            Not to be distributed or copied.
```

```
%
% TIMSHOW is a flexible function for displaying multiple images simultaneously
%         with contrast and colourmap specifications. Contrast and colourmaps
%         apply to all images. Images are spaced as tightly as possible, with
%         a fraction of the figure size padded between images.
%
% Input arguments: (any order)
%    images(s) - any number of 2D grayscale or colour volumes. Images will
%                be rendered in the order they are presented, top to bottom,
%                left to right.
%              * The x-dimension of any image should not have a size of 3,
%                else it will be confused for a colourmap.
%
%    padval    - decimal value on the interval (0, 0.5) dictating the relative
%                padded spacing between images.
%                Default: 0.005
%
%    minmax    - minmax specification for contrast scaling, as in imshow(I,[]).
%                array of size: 1 by 2, or a null: []
%                Default: []
%
%    colourmap - colourmap used for displaying each frame:
%                array of size: M by 3 or a colourmap function
%                Default: curent default figure colormap
%
%              * if 2+ non-image arguments are given, only the last one is used.
%


function [] = timshow(varargin)
[data] = parseargs(varargin);
[data] = initaxes(data);
[data] = showims(data);

function [data] = parseargs(vargs)
data.img = [];
% handle input arguments based on their dimensions
for v = 1:numel(vargs)
    sizev = size(vargs{v});
    if (numel(sizev) == 2) && (all(sizev == [1,1])) && (vargs{v} < 0.5) % pad
        data.pad = vargs{v};
    elseif sizev(2) == 3  % colourmap
        data.colourmap = vargs{v};
    elseif (numel(sizev) == 2) && (all(sizev == [1,2])) % min-max values
        data.minmax = vargs{v};
    elseif sizev(1) == 0 % min-max is []
        data.minmax = [];
    elseif (numel(sizev) == 2) || (numel(sizev) == 3 && sizev(3) == 3) % image
        data.img(end+1).data  = vargs{v};
        data.img(end).size    = size(data.img(end).data);
    else                       % argument not recognized: ignoring
        warning(['Ignoring argument number ',num2str(v),'.']);
    end
end
% default min-max values (adaptive)
if ~isfield(data,'minmax')
    data.minmax = [];
end
% default colourmap
if ~isfield(data,'colourmap')
    data.colourmap = get(0,'defaultfigurecolormap');
end
% default padding
if ~isfield(data,'pad')
    data.pad = 0.005;
```

```
end

function [data] = initaxes(data)
% optimize frame display grid
data.N = numel(data.img);
if data.N == 3
    data.nSubx = 3;
    data.nSuby = 1;
else
    data.nSubx = ceil(sqrt(data.N));
    data.nSuby = ceil(data.N/data.nSubx);
end
% subplot data initialization
for a = 1:data.N
    data.ax(a) = subplot(data.nSuby,data.nSubx,a);
end
% optimize figure display size for the current monitor
screensize = get(0,'screensize');
aspect     = (size(data.img(1).data,1) / size(data.img(1).data,2));
imgSize = min(800, (0.4*screensize(3)) / data.nSubx);
set(gcf,'color','k','position',...
    [(screensize(3) - (imgSize*data.nSubx))/2,...
     (screensize(4) - (imgSize*data.nSuby))/2,...
     (imgSize*data.nSubx),...
     (imgSize*data.nSuby*aspect)]);

function [data] = showims(data)
% show the images in default locations
for i = 1:data.N
    imshow(data.img(i).data,data.minmax,'parent',data.ax(i));
end
% set the positions of the axes
% (must be done after due to disappearing images if they overlap)
for i = 1:data.N
    y = ceil(i / data.nSubx);
    x = mod(i, data.nSubx);
    x(~x) = data.nSubx;
    set(data.ax(i),'position',[(x - 1) / data.nSubx + 0.5*data.pad,  ...
                               1 - (y / data.nSuby - 0.5*data.pad), ...
                                    1 / data.nSubx - data.pad,      ...
                                    1 / data.nSuby - data.pad]);
end
colormap(data.colourmap);
```

8.10 Classification Performance

8.10.1 Pipeline

```
function [NB,Yp,Yploo] = validateclassify(F, Y)
% Full Data
NB  = NaiveBayes.fit(F,Y);
Yp  = NB.predict(F);

% Leave One Out Validatoin
for p = 1:numel(Y)
    Fx = F;  Fx(p,:) = [];
    Yx = Y;  Yx(p)   = [];
    NB = NaiveBayes.fit(Fx,Yx);
    Yploo(p) = NB.predict(F(p,:));
end

% Stats
```

```
fprintf('FULL DATA\n');
printmetrics(Y, Yp)
fprintf('LEAVE ONE OUT\n');
printmetrics(Y, Yploo)

% 3D Plot
plotfeatsandsurface(F,Y,Yp)

function [] = printmetrics(Y, Yp)
M = performanceclassify(Y, Yp, {'acc','sen','spe'});
fprintf(['ACC = ',num2str(M(1),'%02.02f'),'\n',...
         'SEN = ',num2str(M(2),'%02.02f'),'\n',...
         'SPE = ',num2str(M(3),'%02.02f'),'\n']);

function [] = plotfeatsandsurface(F,Y,Yp)
% Feature Names
featnames = {'Interstitial Space Feature',...
             'Nuclear Object Count',...
             'Nuclear Texture Feature'};

% Whiten feature space?
iswhiten = 1;
if iswhiten
    for f = 1:numel(featnames)
        featnames{f} = ['Normalized ',featnames{f}];
        F(:,f) = (F(:,f) - mean(F(:,f))) ./ std(F(:,f));
    end
end
NB = NaiveBayes.fit(F,Y);

% Plotting
fntsz = 12;
hold on;
plotfeatdata( F,  Y,  0,  60); % Truth data
plotfeatdata( F, Yp,  1, 150); % Estimated data
plotsurface(NB, F);            % NB decision surface

% Pretty
xlabel(featnames{1},'fontsize',fntsz);
ylabel(featnames{2},'fontsize',fntsz);
zlabel(featnames{3},'fontsize',fntsz);
set(gca,'fontsize',fntsz);
set(gcf,'position',[100,100,1400,700]);
set(gca,'xgrid','on','ygrid','on','zgrid','on');

% Example Legend
examplelegend();

function [] = examplelegend()
figure;
hold on;

LW = get(0,'defaultLineLineWidth');
set(0,'defaultLineLineWidth', 2);

plot(1,1,'.','color',[0.0,0.5,1.0],'markersize',30);
plot(1,1,'.','color',[1.0,0.2,0.0],'markersize',30);
plot(1,1,'o','color',[0.0,0.5,1.0],'markersize',12);
plot(1,1,'o','color',[1.0,0.2,0.0],'markersize',12);

legend({'True Benign',...
```

```
        'True Malignant',...
        'Predicted Benign',...
        'Predicted Malignant'});
set(gca,'fontsize',14);

set(0,'defaultLineLineWidth', LW);
```

8.10.2 Plotting

```
function [] = plotfeatdata(F, Y, ismodel, psize)
LW = get(0,'defaultLineLineWidth');
set(0,'defaultLineLineWidth', 2);
% Plot the feature space data (representing image instances)
clrset = [0.0, 0.5, 1.0;
          1.0, 0.2, 0.0];
clrs = zeros(size(F));
for k = 1:numel(Y)
    clrs(k,:) = clrset(Y(k)+1,:);
end

if ismodel % estimates
    scatter3(F(:,1),F(:,2),F(:,3), psize*ones(size(Y)), clrs, 'linewidth',2);
else       % truth
    scatter3(F(:,1),F(:,2),F(:,3), psize*ones(size(Y)), clrs, 'filled');
end
set(0,'defaultLineLineWidth', LW);


function [] = plotsurface(NB, F)
% Plot the Naive Bayes decision surface
N   = 50;  % number of surface points (warning: computation is N^3!)
pad = 0.5; % extend the surface by this proportion of the data range

f1 = linspace(min(F(:,1))-pad*range(F(:,1)), max(F(:,1))+pad*range(F(:,1)), N);
f2 = linspace(min(F(:,2))-pad*range(F(:,2)), max(F(:,2))+pad*range(F(:,2)), N);
f3 = linspace(min(F(:,3))-pad*range(F(:,3)), max(F(:,3))+pad*range(F(:,3)), N);

[fx,fy,fz] = meshgrid(f1,f2,f3);
fx = reshape(fx,[N^3,1]);
fy = reshape(fy,[N^3,1]);
fz = reshape(fz,[N^3,1]);

Fs = [fx,fy,fz];     % vectorize features
Ys = NB.predict(Fs); % vector solution

V = smooth3(reshape(Ys,[N,N,N]),'box',9); % reshape to volume, smooth
P = patch(isosurface(f1,f2,f3,V));        % render the surface
set(P,'FaceColor','k','EdgeColor','none','FaceAlpha',0.4); % pretty
camlight; % very pretty
```

8.10.3 Performance Metrics

```
function [M, TP_FP_TN_FN] = performanceclassify(y, yp, metric)

TP_FP_TN_FN = confusion(y,yp);

for m = 1:numel(metric)
    switch metric{m}
        case 'acc'
            metricfcn = @accuracy;
```

```
        case 'sen'
            metricfcn = @sensitivity;
        case 'spe'
            metricfcn = @specificity;
        case 'ppv'
            metricfcn = @pospredvalue;
        case 'npv'
            metricfcn = @negpredvalue;
        case 'fpr'
            metricfcn = @falseposrate;
        case 'fnr'
            metricfcn = @falsenegrate;
        otherwise
            error('Metric not recognized.');
    end
    M(m) = metricfcn(TP_FP_TN_FN);
end
function [TP_FP_TN_FN] = confusion(y,yp)
TP_FP_TN_FN(1) = sum( (y(:) == 1) & (yp(:) == 1) );
TP_FP_TN_FN(2) = sum( (y(:) == 0) & (yp(:) == 1) );
TP_FP_TN_FN(3) = sum( (y(:) == 0) & (yp(:) == 0) );
TP_FP_TN_FN(4) = sum( (y(:) == 1) & (yp(:) == 0) );

function [acc] = accuracy(TP_FP_TN_FN)
TP = TP_FP_TN_FN(1);
TN = TP_FP_TN_FN(3);
acc = (TP + TN) / sum(TP_FP_TN_FN);
function [sen] = sensitivity(TP_FP_TN_FN)
TP = TP_FP_TN_FN(1);
FN = TP_FP_TN_FN(4);
sen = TP / (TP + FN);
function [spe] = specificity(TP_FP_TN_FN)
FP = TP_FP_TN_FN(2);
TN = TP_FP_TN_FN(3);
spe = TN / (TN + FP);
function [ppv] = pospredvalue(TP_FP_TN_FN)
TP = TP_FP_TN_FN(1);
FP = TP_FP_TN_FN(2);
ppv = TP / (TP + FP);
function [npv] = negpredvalue(TP_FP_TN_FN)
TN = TP_FP_TN_FN(3);
FN = TP_FP_TN_FN(4);
npv = TN / (TN + FN);
function [fpr] = falseposrate(TP_FP_TN_FN)
FP = TP_FP_TN_FN(2);
TN = TP_FP_TN_FN(3);
fpr = FP / (FP + TN);
function [fnr] = falsenegrate(TP_FP_TN_FN)
FN = TP_FP_TN_FN(4);
TP = TP_FP_TN_FN(1);
fnr = FN / (FN + TP);
```

8.11 Segmentation Performance

8.11.1 Pipeline & Plotting

```
function [] = validatenucseg(P1)
folder   = 'nucsegvalidate/';

% truth and estimate
gt  = imread([folder,P1.fname(1:end-4),'_gt.tif']);
est = P1.nucseg;
```

```
% truth is colour image, need to collapse to binary
gt = 1 - (mean(gt(:,:,3),3))./255;
gt(gt ~= 1) = 0;

% zoom indexes
zoomy = 1:300;
zoomx = 1:300;

timshow(P1.img(zoomy,zoomx,:), (2*gt(zoomy,zoomx) + single(est(zoomy,zoomx))), hot);

% statistics
[M] = performancenucseg(gt,est,'DSC');
fprintf(['DSC = ',num2str(M,'%02.02f'),'\n']);
[M] = performancenucseg(gt,est,'sen');
fprintf(['SEN = ',num2str(M,'%02.02f'),'\n']);
[M] = performancenucseg(gt,est,'spe');
fprintf(['SPE = ',num2str(M,'%02.02f'),'\n']);
```

8.11.2 Performance Metrics

```
function [M,Mc] = performancenucseg(GT,S,metric)
classvals = unique(GT);
classvals(classvals == 0) = [];
TP_FP_TN_FN = zeros(1,4);

switch metric
    case 'DSC'
        metricfcn = @dicesimilarity;
    case 'OF'
        metricfcn = @overlapfactor;
    case 'EF'
        metricfcn = @extrafraction;
    case 'sen'
        metricfcn = @sensitivity;
    case 'spe'
        metricfcn = @specificity;
    otherwise
        error('Metric not recognized.');
end

for c = 1:numel(classvals)
    GTC = uint8(GT == classvals(c));
    SC  = uint8(S  == classvals(c));

    TP_FP_TN_FNc = segclassrates(GTC,SC);
    Mc(c) = metricfcn(TP_FP_TN_FNc);
    TP_FP_TN_FN  = TP_FP_TN_FN + TP_FP_TN_FNc;
end
M = metricfcn(TP_FP_TN_FN);

function [TP_FP_TN_FN] = segclassrates(GTC,SC)
TP_FP_TN_FN(1) = sum( (GTC(:) == 1) & (SC(:) == 1) );
TP_FP_TN_FN(2) = sum( (GTC(:) == 0) & (SC(:) == 1) );
TP_FP_TN_FN(3) = sum( (GTC(:) == 0) & (SC(:) == 0) );
TP_FP_TN_FN(4) = sum( (GTC(:) == 1) & (SC(:) == 0) );
function [DSC] = dicesimilarity(TP_FP_TN_FN)
TP = TP_FP_TN_FN(1);
FP = TP_FP_TN_FN(2);
FN = TP_FP_TN_FN(4);
DSC = (2*TP) / (2*TP + FP + FN);
```

```
function [OF] = overlapfactor(TP_FP_TN_FN)
TP = TP_FP_TN_FN(1);
FN = TP_FP_TN_FN(4);
OF = TP / (TP + FN);
function [EF] = extrafraction(TP_FP_TN_FN)
TP = TP_FP_TN_FN(1);
FP = TP_FP_TN_FN(2);
FN = TP_FP_TN_FN(4);
EF = FP / (TP + FN);
function [sen] = sensitivity(TP_FP_TN_FN)
TP = TP_FP_TN_FN(1);
FN = TP_FP_TN_FN(4);
sen = TP / (TP + FN);
function [spe] = specificity(TP_FP_TN_FN)
FP = TP_FP_TN_FN(2);
TN = TP_FP_TN_FN(3);
spe = TN / (TN + FP);
```