ABSTRACT

A Beefy Frangi Filter for Noisy Vascular Segmentation and Network Connection in PCSVN

By

Lucas Wukmer

December 2018

Recent statistical analysis of placental features has suggested the usefulness of studying key features of the placental chorionic surface vascular network (PCSVN) as a measure of overall neonatal health. A recent study has suggested that reliable reporting of these features may be useful in identifying risks of certain neurodevelopmental disorders at birth. The necessary features can be extracted from an accurate tracing of the surface vascular network, but such tracings must still be done manually, with significant user intervention. Automating this procedure would not only allow more data acquisition to study the potential effects of placental health on later conditions, but may ideally serve as a real-time diagnostic for neonatal risk factors as well.

Much work has been to develop reliable vascular extraction methods for well-known image domains (such as retinal MRA images) using Hessian-based filters, namely the (multiscale) Frangi filter. It is desirable to extend these arguments to placental images, but this approach is greatly hindered by the inherent irregularity of the placental surface as a whole, which introduces significant noise into the image domain. A recent attempt was made to apply an additional local curvilinear filter to the Frangi result in an effort to remove some noise from the final extraction.

Here we propose an alternate extraction method. First, we use arguments from

Frangi's original paper to provide a proper selection of parameters for our particular image domain. Using the same arguments from differential geometry that gave rise to the Frangi filter, we calculate the leading principal direction (eigenvector of the Hessian) to indicate the directionality of curvilinear features at a particular scale. We are then able to apply an appropriately-oriented morphological filter to our Frangi targets at select scales to remove noise. This approach differs significantly from previous efforts in that morphological filtering will take place at each scale space, rather than being performed one time following multiscale synthesis. Noise removal performed in this way is expected to aide in coherent interpretation of targets that should appear in a connected network.

Finally, we discuss an important advancement in implementation–scale space conversion for differentiation (i.e. gaussian blur) via Fast Fourier Transform (FFT) rather than a more traditional convolution with a gaussian kernel, which offers a significant speedup. This thesis will also contain a general, in depth summary of both multiscale Hessian filters and scale-space theory.

We demonstrate the effectiveness of our improved vascular extraction technique on several of the following image domains: a private database of barium-injected samples provided by University of Rochester, uninjected/raw placental samples from Placental Analytics LLC, a collection of simulated images, the DRIVE and STARE databases of retinal MRAs, and a new collection of computer-generated images with significant curvilinear content.

Time permitting, this research will be extended to include a method of network connection, so that a logically connected vascular network is realized (i.e. network completion).

A Beefy Frangi Filter for Noisy Vascular Segmentation and Network Connection in PCSVN

A THESIS

Presented to the Department of Department of Mathematics and Statistics

California State University, Long Beach

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Applied Mathematics

Committee Members:

Jen-Mei Chang, Ph.D. (Chair)
James von Brecht, Ph.D.
William Ziemer, Ph.D.

College Designee:

Tangan Gao, Ph.D.

By Lucas Wukmer

B.S., 2013, University of California, Los Angeles

December 2018

WE, THE UNDERSIGNED MEMBERS OF THE COMMITTEE,

HAVE APPROVED THIS THESIS

A Beefy Frangi Filter for Noisy Vascular Segmentation and Network Connection in PCSVN

By

Lucas Wukmer

COMMITTEE MEMBERS

Jen-Mei Chang, Ph.D. (Chair)	Mathematics and Statistics
James von Brecht, Ph.D.	Mathematics and Statistics
William Ziemer, Ph.D.	Mathematics and Statistics

ACCEPTED AND APPROVED ON BEHALF OF THE UNIVERSITY

Tangan Gao, Ph.D.
Department Chair, Mathematics and Statistics

California State University, Long Beach

December 2018

ACKNOWLEDGEMENTS

Acknowledgments go here.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The Applied Problem

From [cite salafia], it is useful to develop a neonatal test for high risk of Autism Spectrum Disorder. There is some evidence as in [6] that there is some correlation between risk and placental health. Most ASD cases are not diagnosed until the child reaches three or four, so the benefit of any neonatal testing would be very beneficial, as the brain may be more receptive to treatment at a young age. In particular, it was shown in [6] that measurements of the placental chorionic surface vascular network (PCSVN) may be useful in identifying such risk. Whereas previous studies have required manual tracing of the PCSVN in order to make these measurements, there was been work to automate this procedure, as in [13] [8], and so on. We continue the work of developing a procedure to automate extraction of the PCSVN.

Our basic goal of ”vascular network extraction” is a frequent one in image processing. There are have been many techniques adapted to extracting vascular networks. The placenta in particular presents a greater degree of difficulty due to the nature of the vascular network. It’s a surface network and the ”background” has a great degree of topology itself, causing many naive approaches to fail that work with other image domains.

Research Goals

Our present work is to improve upon the vascular network extraction developed in [13] and test our extraction against the manual traces developed.

CHAPTER 2

MATHEMATICAL METHODS

Our goal is establish a resource efficient method of finding curvilinear content in 2D grayscale digital images using concepts of differential geometry. We proceed by (i) establishing a standard method of viewing these images as 2D surfaces, (ii) developing a minimal yet rigorous distillation of differential geometry to obtain suitiable quantifiers for the study of curvilinear structure in 3D surfaces, (iii) establishing a filter based on these quantifiers, and finally (iv) developing methods necessary for efficient computation of the filter.

Problem Setup in Image Processing

A digital 2D grayscale image is given by a *MxN* array of pixels, whose intensity is given by an integer value between 0 and 255.

Definition 2.1 (Image as a pixel matrix)**.**

$$\mathtt{I} \in \mathbb{N}^{M\times N} \quad with \quad 0 \leq \mathtt{I}_{ij} \leq 2^8 - 1$$

For theoretical purposes, we wish to consider any such picture to ultimately be a sampling of a 2D continuous surface. We also require that this surface is sufficiently continuous as to admit the existence of second partial derivatives.

Definition 2.2 (Image as an interpolated surface)**.**

$$h : \mathbb{R}^2 \to \mathbb{R} \quad with \quad h \in C^2\left(\mathbb{R}^2\right), \quad where \quad h(i,j) = \mathtt{I}_{ij}\ \forall (i,j) \in \{0,...,M\} \times \{0,...,N\} \subset \mathbb{N}^2$$

That is, the function h is identical to the pixel matrix $\mathtt{I}$ at all integer inputs, and simply a "smooth enough" interpolation of those points for all other values.

It is of course necessary to admit that `I` is not really a perfect representation of the underlying "content" within the picture. Not only is information lost when `I` is stored as an integer, there are also elements of noise and anomalies of lighting that would constitute noise to the original signal. There are multiple treatments of image processing that do address this discrepancy in a pragmatic way [11], especially when the goal is noise reduction. However, we will be content to simply represent the pixels of `I` as the ultimate "cause" of the surface h in definition 2.2, and worry not about how faithfully that sampling corresponds to the real world. Moreover, though our samples in the image domain have been carefully prepared (as outlined in), there are numerous shortcomings therein, and improvements to the veracity of our original signal could be made from many angles. Though we shall draw upon the notion of the pixel matrix `I` as a sampling again to motivate our development of scale space theory in section 2.4, we ultimately use these techniques because we find them successful to our problem.

Differential Geometry

We wish to describe the structure of an image as a surface. To do this, we develop the notion of curvature of a surface in $\mathbb{R}^3$ in a standard way, following [17] (although any undergraduate text in Differential Geometry should prove satisfactory).

Preliminaries of Differential Geometry

Given an open subset $U \subset \mathbb{R}^2$ and a twice differentiable function $h : U \to \mathbb{R}$ (as in definition 2.2) we define the graph, f, of h in the following definition.

Definition 2.3. *The surface f is a graph (of the function h) when*

$$f : U \to \mathbb{R}^3 \quad by \quad f(u_1,u_2) \;=\; \Big(u_1,u_2,h(u_1,u_2)\Big)\,, \quad u = (u_1,u_2) \in U \subset \mathbb{R}^2$$

Since the graph f is clearly one-to-one by definition, we may readily associate any input $u \in U$ with its corresponding output $p \in f[U]$, i.e.
$p = f(u) = f(u_1,u_2) \;=\; \Big(u_1,u_2,h(u_1,u_2)\Big)$, depending on whether we wish to focus on a

point of a graph in terms of its input or in terms of the structure of the graph itself.

Our development of curvature ultimately will hinge upon a careful consideration of the tangent plane of f at a point p, for we will require a concrete definition of both the tangent space within the domain and image of f, as well as the so called "differential" of f, the lattermost of which we will only define for the immediate case required Seeing that f is one-to-one should make a lot of this futzing about complete overkill, but I've yet to find a way to distill it. That is, this development works for any parametrized surface element, not necessarily a graph. Whatever for now.

Definition 2.4 (Tangent space of U at u)**.**

$$T_u U = \{u\} \times \mathbb{R}^2$$

Definition 2.5 (Tangent space of $\mathbb{R}^3$ at p)**.**

$$T_p \mathbb{R}^3 = \{p\} \times \mathbb{R}^3$$

It is immediately clear that $T_u U$ and $T_p \mathbb{R}^3$ are isomorphic to $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively, and we can easily visualize elements of $T_u U$ are tangent vectors in $\mathbb{R}^2$ "originating" at the point u, and elements of $T_p \mathbb{R}^3$ are tangent vectors "originating" at the point p.

Definition 2.6 (The differential of f at a point u)**.** *$Df|_u$ is the map from $T_u U$ into $\mathbb{R}^3$ given by*

$$Df|_u : T_u U \to T_{f(u)} \mathbb{R}^3 \quad \text{by} \quad w \mapsto J_f(u) \cdot v$$

where $J_f(u)$ is the Jacobian of f evaluated at some fixed point $u \in U$, i.e. the matrix

$$J_f(u) = \left[\left. \frac{\partial f_i}{\partial u_j} \right|_u \right]_{i,j}$$

Although not necessary presently, we could just as easily consider the differential of an arbitrary function as a map between tangent vectors in the function's domain and

tangent vectors in its range. We could also just identify this as mapping $U \to \mathbb{R}^3$ by the obvious isomorphism described above. and then differential of f at x is simply a linear transformation of between the tangent spaces $T_u U$ and $T_p\mathbb{R}^3$ where the transformation in question is given by the Jacobian. We can define such a differential at any point u in the domain.

With these three definitions, we are equipped to give a formal definition of $T_u f$, the tangent plane of f at an input u.

Definition 2.7 (Tangent plane of a graph)**.**

$$T_u f := Df|_u(T_u U) \subset T_{f(u)}\mathbb{R}^3 = T_p\mathbb{R}^3$$

This vectors of this plane can thus be identified as tangent vectors from $T_u U$ that have been passed through the differential mapping $Df|_u$. We shall denote a generic tangent vector $X \in T_u f$ at point p. We may expand any such vector X in terms of the basis $\left\{\frac{\partial f}{\partial u_i}\right\}_{i=1,2}$; that is, $\text{span}\left\{\frac{\partial f}{\partial u_1}, \frac{\partial f}{\partial u_2}\right\} = T_u f$.

Given the level of abstraction above, it may be refreshing to explicitly show the linear independence of this set in the case of an arbitrary graph f.

Lemma 2.1. *When f is a graph, for all points $u \in U$, $\left\{\frac{\partial f}{\partial u_1}, \frac{\partial f}{\partial u_2}\right\}$ is in fact a basis for the tangent plane $T_u f$.*

Quite obviously, we're assuming $(1,0),(0,1) \in U$. If this is not the case, we pick some α small enough so that $(\alpha,0)$ and $(0,\alpha)$ are contained and this scaled version would serve as a basis instead.

Proof. Given the definition of a graph f as in definition 2.3, we can directly calculate the partial derivatives of f at a point u.

$$f_{u_1} = (1,0,h_{u_1}(u)) \quad \text{and} \quad f_{u_2} = (0,1,h_{u_2}(u))$$

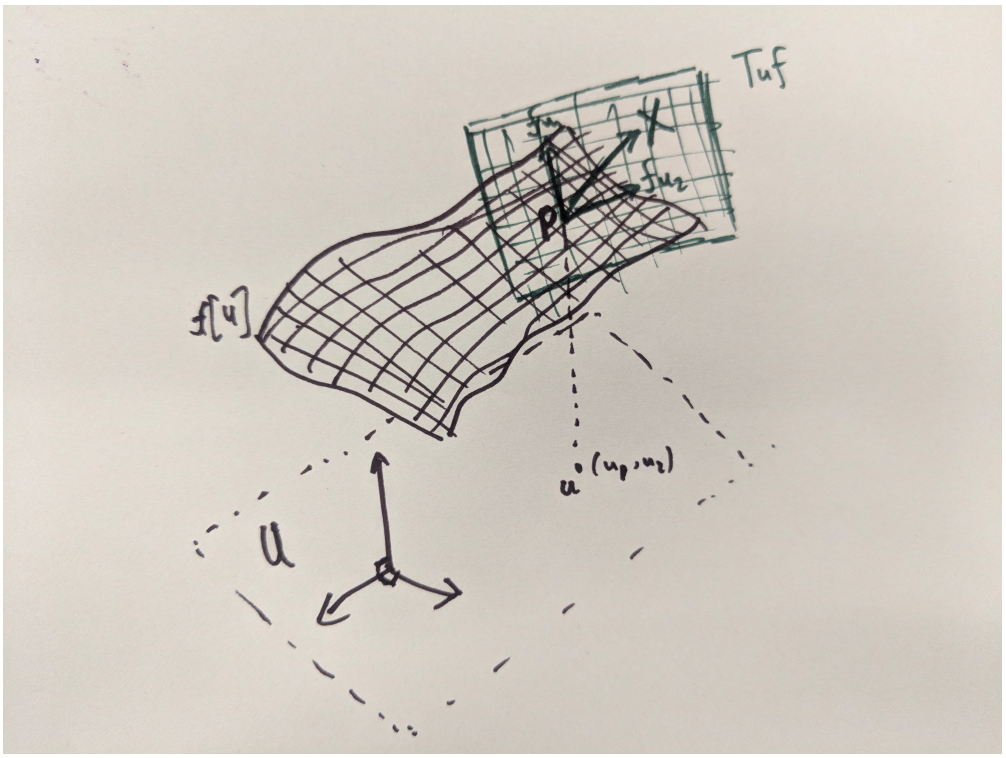


FIGURE 1: Tangent plane of a graph

which are obviously linearly independent. Then $Df|_u(1,0) = f_{u_1}$, and $Df|_u(0,1) = f_{u_2}$, which shows $\left\{\frac{\partial f}{\partial u_1}, \frac{\partial f}{\partial u_2}\right\} \in T_u f$. Thus $\left\{\frac{\partial f}{\partial u_1}, \frac{\partial f}{\partial u_2}\right\}$ is a linearly independent subset of $T_u f$, and can serve as its basis. □

The partials derivatives of f are not, in general, orthogonal at any point u, unless it happens that h_{u_1} or h_{u_2} is zero. A visualization of some of the above is given in fig. 1, although note that f_{u_1} and f_{u_2} accidentally appear orthogonal.

We now concern ourselves with developing the notion of curvature on a surface. First, we need to consider an arbitrary regular curve (i.e. differentiable, one-to-one, non-zero derivative) contained within the image of f.

Curvature of a surface and its calculation

In the context of a regular arc-length parametrized curve $c : I \to \mathbb{R}^3$ parametrized along some closed interval $I \in \mathbb{R}$ (that is, a differentiable, one-to-one curve where

$c'(s) = 1 \ \ \forall s \in I$), curvature at a point $s \in I$ is defined simply as the magnitude of the curve's acceleration: $\kappa(s) := \|c''(s)\|$.

To extend the notion of curvature of a surface f, we can consider the curvature of such an arbitrary curve embedded within the surface.

Definition 2.8 (Surface curve)**.** *Given a closed interval $I \subset \mathbb{R}$, we call the regular curve $c : I \to \mathbb{R}^3$ a surface curve in the event that* $\text{image}(c) \subset \text{image}(f)$ *entirely. The one-to-one-ness of the graph f ensures that we can define (for the given curve) an intermediary parametrization θ_c so that $c = f \circ \theta_c$. That is,*

$$\theta_c : I \to U \text{ by } \theta(t) = \Big(\theta_1(t), \theta_2(t)\Big)$$

so that $c(t) = f(\theta_c(t)) \ \forall t \in I$, and $c[I] = f\left[\theta_c[I]\right]$..

Note as well that the velocity of this particular curve lies within $T_u f$. This can be seen by an elementary application of chain rule:

$$-\frac{dc}{dt} = -\frac{d}{dt}\Big[f(\theta_c(t))\Big] \tag{2.1}$$

$$= -\frac{d}{dt}\Big[f(\theta_1(t), \theta_2(t))\Big] \tag{2.2}$$

$$= \theta_1'(t)\left(\frac{\partial f}{\partial u_1}\right) + \theta_2'(t)\left(\frac{\partial f}{\partial u_2}\right) \in T_u f. \tag{2.3}$$

Considering a point $p \in I$ and its associated point $u = \theta_c(p)$, we wish to compare the curvatures of all (regular) surface curves passing through the point p at some particular velocity.

We now present a main result that provides a notion of curvature of a surface.

Theorem 2.2 (Theorem of Meusnier)**.** *Given a point $u \in U$ and a tangent direction $X \in T_u f$, any regular curve on the surface $c : I \to \text{image}(f)$ with $p \in I : \theta_c(p) = u$ where $c'(p) = X$ will have the same curvature.*

In other words, any two curves on the surface with a common velocity at a given point on the surface will have the same curvature. To prove this, we'll require one final definition.

Definition 2.9 (The Gauss Map)**.** *The Gauss map at a point $p = f(u)$ is the unit normal to the tangent plane*

$$\nu : U \to \mathbb{R}^3 \quad by \quad \nu(u) := \frac{\frac{\partial f}{\partial u_1} \times \frac{\partial f}{\partial u_2}}{\| \frac{\partial f}{\partial u_1} \times \frac{\partial f}{\partial u_2} \|}$$

Each partial above understood to be evaluated at the input $u \in U$; that is, we calculate $\frac{\partial f}{\partial u_i}\Big|_u$. The existence of the cross product in its definition makes it clear that $\nu \perp \frac{\partial f}{\partial u_i}$ each $i = 1,2$. A simple dimensionality argument of $\mathbb{R}^3$ implies that these must exist in $T_u f$. However, we can also show it directly:

To show that $\left\{\frac{\partial \nu}{\partial u_1}, \frac{\partial \nu}{\partial u_2}\right\} \in T_u f$, first note that at any particular $u \in U$, $\langle \nu, \nu \rangle = 1 \implies \frac{\partial}{\partial u_i}\langle \nu, \nu \rangle = 0$, and so by chain rule $2\langle \frac{\partial \nu}{\partial u_i}, \nu \rangle = 0 \implies \frac{\partial \nu}{\partial u_i} \perp \nu$. Since $\nu \perp \text{span}\left\{\frac{\partial f}{\partial u_i}\right\}$ as well (since ν its outer product), in $\mathbb{R}^3$, this implies $\text{span}\left\{\frac{\partial \nu}{\partial u_i}\right\} \parallel \text{span}\left\{\frac{\partial f}{\partial u_i}\right\}$.

Thus, we have $\text{span}\left\{\frac{\partial \nu}{\partial u_1}, \frac{\partial \nu}{\partial u_2}\right\} \subset T_u f$ as well and we can also use it as a basis.

We are finally ready to prove theorem 2.2, the Theorem of Meusnier.

Proof. Let $X \in T_u f$ be given and consider some curve where $\frac{\partial c}{\partial t}(u) = X$ where $X \in T_u f$. We wish to decompose the curve's acceleration along the orthogonal vectors X and the Gauss map $\nu = \nu(u_1, u_2) = \frac{\frac{\partial f}{\partial u_1} \times \frac{\partial f}{\partial u_2}}{\| \frac{\partial f}{\partial u_1} \times \frac{\partial f}{\partial u_2} \|}$ as in definition 2.9. Note that X and ν are indeed orthogonal, as $X \in \text{span}\left\{\frac{\partial f}{\partial u_i}\right\} = T_u f$, and $\nu \perp T_u f$). We then have (at this fixed point $u = \theta_c(p)$)

$$c'' = \langle c'', X \rangle X + \langle c'', \nu \rangle \nu \tag{2.4}$$

.

Because c is a regular curve, we either have $c'' = 0$, or $c' \perp c''$, since $\|c'\| = 1$ implies $0 = \frac{d}{dt}\langle c', c' \rangle = 2\langle c'', c' \rangle$. Thus

$$\langle c'', X\rangle = \langle c'', c'\rangle = 0$$

and we can rewrite the second coefficient of eq. (2.4) using the chain rule:

$$\langle c'', v\rangle = \frac{\partial}{\partial t}[\langle c', v\rangle] - \langle c', \frac{\partial v}{\partial t}\rangle \tag{2.5}$$
$$= \frac{\partial}{\partial t}[\langle X, v\rangle] - \langle c', \frac{\partial v}{\partial t}\rangle \tag{2.6}$$
$$= 0 - \langle X, \frac{\partial v}{\partial t}\rangle \tag{2.7}$$

Thus, we can express the curvature at this point on our selected curve as

$$\|c''\| = \|\langle c'', X\rangle X + \langle c'', v\rangle v\| = \|0 + \langle c'', v\rangle v\| \tag{2.8}$$
$$= -\langle X, \frac{\partial v}{\partial t}\rangle \|v\| \tag{2.9}$$
$$= -\langle X, \frac{\partial v}{\partial t}\rangle \tag{2.10}$$
$$= \langle X, -\frac{\partial v}{\partial t}\rangle \tag{2.11}$$

We may compute the quantity $-\frac{\partial v}{\partial t}$ that appears in eq. (2.11) via chain rule:

$$-\frac{dv}{dt} = -\frac{d}{dt}\Big[v(u_1, u_2)\Big] \tag{2.12}$$
$$= -\frac{d}{dt}\Big[v(\theta_1(t), \theta_2(t))\Big] \tag{2.13}$$
$$= \theta_1'(t)\left(-\frac{\partial v}{\partial u_1}\right) + \theta_2'(t)\left(-\frac{\partial v}{\partial u_2}\right) \tag{2.14}$$

Identifying span$\left\{-\frac{\partial v}{\partial u_i}\right\}_{i=1,2}$ as a subset of $T_u f$, we can define a linear transformation L which maps the basis $\left\{\frac{\partial f}{\partial u_i}\right\}_{i=1,2}$ to this subset:

Definition 2.10 (The Weingarten Map)**.**

$$\mathsf{L} : T_u f \to T_u f \quad \textit{given by the composition} \quad \mathsf{L} = Dv \circ (Df)^{-1}.$$

That is, $\mathsf{L}(\frac{\partial f}{\partial u_i}) = -\frac{\partial v}{\partial u_i}$ for $i = 1, 2$, where the negative sign comes about from blind adherence to eq. (2.14) and eq. (2.11). We will This allows us to rewrite the time

derivative of the Gauss map eq. (2.12) as

$$-\frac{dv}{dt} = \theta_1'(t)\left(-\frac{\partial v}{\partial u_1}\right) + \theta_2'(t)\left(-\frac{\partial v}{\partial u_2}\right) \tag{2.15}$$

$$= \theta_1'(t)\left(\mathsf{L}\left(\frac{\partial f}{\partial u_1}\right)\right) + \theta_2'(t)\left(\mathsf{L}\left(\frac{\partial f}{\partial u_2}\right)\right) \tag{2.16}$$

$$= \mathsf{L}\left[\theta_1'(t)\left(\frac{\partial f}{\partial u_1}\right) + \theta_2'(t)\left(\frac{\partial f}{\partial u_2}\right)\right] \tag{2.17}$$

$$= \mathsf{L}\left(\frac{d}{dt}\left[f\left(\theta(t)\right)\right]\right) = \mathsf{L}\left(\frac{d}{dt}\left[c(t)\right]\right) = \mathsf{L}(X) \tag{2.18}$$

With this, we can re-express the curvature of our curve from eq. (2.11) as the much simpler

$$\|c''\| = \langle X, -\frac{\partial v}{\partial t}\rangle = \langle X, \mathsf{L}(X)\rangle \tag{2.19}$$

The linear transformation L from definition 2.10, and thereby the computation of curvature given in eq. (2.19), depends only on the point u and the selected direction X, not on the particular curve c at all. □

To recap, given a point u on the surface and an arbitrary vector X in the tangent plane, we can calculate the curvature of any surface curve with velocity X there. In fact, we refer to this intrinsic quantity as the normal curvature of the surface.

Definition 2.11. *The normal curvature of a surface, denoted κ_v at point u in the direction X is given by*

$$\kappa_v := \langle X, \mathsf{L}(X)\rangle$$

In fact, theorem 2.2 shows that the normal curvature is an intrinsic property of the surface–it depends only on the surface at a point, and no reference to any particular curve on the surface is necessary or implied.

The map L introduced in the proof above is known as the Weingarten map and is implicitly defined at each $u \in U$. We wish to make its existence rigorous as well as find a matrix representation for it, using the standard motivation that $\mathsf{L}(\frac{\partial f}{\partial u_i}) = -\frac{\partial v}{\partial u_i}$.

That is, we may trace any $X \in T_u f$ which has been expanded in terms of the basis $\left\{\frac{\partial f}{\partial u_1}, \frac{\partial f}{\partial u_2}\right\}$ and map it to the span of $\left\{-\frac{\partial \nu}{\partial u_1}, -\frac{\partial \nu}{\partial u_2}\right\}$.

The Weingarten map can be formally shown to be well-defined, invariant under coordinate transformation in the general case, which is certainly useful for surfaces f that are not graphs. We refer to [17] for the general proof. The situation is much less delicate if f is a graph–the linear transformation may be simply constructed, and we proceed by simply calculating its matrix representation.

Lemma 2.3. *The Weingarten map as in definition 2.10 is well-defined for graphs.*

To find a matrix representation for L, (which we will denote $\widehat{\mathsf{L}} \in R^{2\times 2}$) we simply wish to find a linear transformation such that $\widehat{\mathsf{L}} \left.\frac{\partial f}{\partial u_i}\right|_{T_u f} = -\left.\frac{\partial \nu}{\partial u_i}\right|_{T_u f}$ for $i = 1, 2$ where $-X|_{T_u f}$ denotes that $X \in T_u f$ is being represented in so-called 'local coordinates' for $T_u f$ (Strictly speaking, of course $T_u f \subset \mathbb{R}^3$ and thus $\frac{\partial f}{\partial u_i} \in \mathbb{R}^3$. Thus when we say $\left.\frac{\partial f}{\partial u_i}\right|_{T_u f}$ we are referring to this 3-vector expanded with respect to the two-dimensional basis for $T_u f$). In matrix form, we describe this situation as

$$\left[\widehat{\mathsf{L}}\right]\begin{bmatrix} | & | \\ \left.\frac{\partial f}{\partial u_1}\right|_{T_u f} & \left.\frac{\partial f}{\partial u_2}\right|_{T_u f} \\ | & | \end{bmatrix} = \begin{bmatrix} | & | \\ \widehat{\mathsf{L}}\left.\frac{\partial f}{\partial u_1}\right|_{T_u f} & \widehat{\mathsf{L}}\left.\frac{\partial f}{\partial u_2}\right|_{T_u f} \\ | & | \end{bmatrix} \tag{2.20}$$

$$= \begin{bmatrix} | & | \\ -\left.\frac{\partial \nu}{\partial u_1}\right|_{T_u f} & -\left.\frac{\partial \nu}{\partial u_2}\right|_{T_u f} \\ | & | \end{bmatrix} \tag{2.21}$$

Now, representing each vector in $T_u f$ with respect to the basis $\left\{\frac{\partial f}{\partial u_i}\right\}$, we have

$$\implies \left[\widehat{\mathsf{L}}\right]\begin{bmatrix} - & \frac{\partial f}{\partial u_1} & - \\ - & \frac{\partial f}{\partial u_2} & - \end{bmatrix}\begin{bmatrix} | & | \\ \frac{\partial f}{\partial u_1} & \frac{\partial f}{\partial u_2} \\ | & | \end{bmatrix} = \begin{bmatrix} - & \frac{\partial f}{\partial u_1} & - \\ - & \frac{\partial f}{\partial u_2} & - \end{bmatrix}\begin{bmatrix} | & | \\ -\frac{\partial \nu}{\partial u_1} & -\frac{\partial \nu}{\partial u_2} \\ | & | \end{bmatrix} \tag{2.22}$$

We can simplify this greatly by defining

$$g_{ij} := \langle \frac{\partial f}{\partial u_i}, \frac{\partial f}{\partial u_j} \rangle \quad \text{and} \quad h_{ij} := \langle \frac{\partial f}{\partial u_i}, -\frac{\partial \nu}{\partial u_j} \rangle \tag{2.23}$$

so that

$$\begin{bmatrix} \widehat{\mathsf{L}} \end{bmatrix} \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \tag{2.24}$$

Then we rearrange to solve for $\widehat{\mathsf{L}}$ as

$$\widehat{\mathsf{L}} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix}^{-1} \tag{2.25}$$

where $\left[g_{ij}\right]$ is clearly invertible, as the set $\left\{\frac{\partial f}{\partial u_j}\right\}$ is linearly independent.

It should be noted that this matrix representation is accurate not only for the surface of a graph, but for any *generalized* surface $f : U \to \mathbb{R}^3$ with $u \mapsto (x(u), y(u), z(u))$ as well. We shall later show that this calculation simplifies (somewhat) in the case that our surface is a graph.

Our final goal to is to characterize such normal curvatures. Namely, we wish to establish a method of determining in which directions an extremal normal curvature occurs.

Principal Curvatures and Principal Directions

To do so, we shall consider the relationship between the direction X and the normal curvature κ_ν in that direction at some specified u.

First, we need the following lemma:

Lemma 2.4. *If $A \in R^{n\times n}$ is a symmetric real matrix, $v \in R^n$ and given the dot product $\langle \cdot, \cdot \rangle$, we have $\nabla_v \langle v, Av \rangle = 2Av$. In particular, when $A = I$ the identity matrix, we have $\nabla_v \langle v, v \rangle = 2v$.*

Proof. The result is uninterestingly obtained by tracking each (the 'ith') component of $\nabla_v \langle v, Av \rangle$:

$$\Big(\nabla_v\langle v, Av\rangle\Big)_i = \frac{\partial}{\partial v_i}\Big[\langle v, Av\rangle\Big] = \frac{\partial}{\partial v_i}\left[\sum_{j=1}^{n} v_j (Av)_j\right] \tag{2.26}$$

$$= \frac{\partial}{\partial v_i}\left[\sum_{j=1}^{n} v_j \sum_{k=1}^{n} a_{jk} v_k\right] \tag{2.27}$$

$$= \frac{\partial}{\partial v_i}\left[a_{ii}v_i^2 + v_i\sum_{k\neq i} a_{ik}v_k + v_i\sum_{j\neq i} a_{ji}v_j + \sum_{j\neq i}\sum_{k\neq i} v_j a_{jk} v_k\right] \tag{2.28}$$

$$= 2a_{ii}v_i + \sum_{k\neq i} a_{ik}v_k + \sum_{j\neq i} a_{ji}v_j + 0 \tag{2.29}$$

$$= 2a_{ii}v_i + 2\sum_{k\neq i} a_{ik}v_k = 2\sum_{k=1}^{n} a_{ik}v_k = 2\Big(Av\Big)_i \tag{2.30}$$

$$\implies \nabla_v\langle v, Av\rangle = 2Av. \tag{2.31}$$

□

We are now ready for the major result of this section, which ties the Weingarten map to the notion of normal curvatures.

Theorem 2.5 (Theorem of Olinde Rodrigues)**.** *Fixing a point $u \in U$, a direction $X \in T_u f$ minimizes the normal curvature $\kappa_\nu = \langle \mathsf{L}X, X\rangle$ subject to $\langle X, X\rangle = 1$ iff X is a (normalized) eigenvector of the Weingarten map L.*

Proof. In the following, we will assume that $X \in T_u f$ is expanded, in local coordinates, i.e. along a two dimensional basis (such as $\left\{\frac{\partial f}{\partial u_i}\right\}_{i=1,2}$) and thus can refer to L freely as the 2×2 matrix $\widehat{\mathsf{L}}$. Using the method of Lagrange multipliers, we define the Lagrangian:

$$\mathscr{L}(X;\lambda) := \langle \widehat{\mathsf{L}}X, X\rangle - \lambda\Big(\langle X, X\rangle - 1\Big) \tag{2.32}$$

Extremal values occur when $\nabla_{X,\lambda}\mathscr{L}(X;\lambda) = 0$, which results in the two equations

$$\begin{cases} \nabla_X\langle \widehat{\mathsf{L}}X, X\rangle - \lambda\nabla_X\left(\langle X, X\rangle - 1\right) = 0 \\ \qquad\qquad\qquad\qquad \langle X, X\rangle - 1 = 0 \end{cases} \tag{2.33}$$

The second requirement is simply the constraint that X is normalized. Using the previous lemma, we can simplify the first result as follows:

$$\begin{aligned} \nabla_X \langle \widehat{\mathsf{L}} X, X \rangle - \lambda \nabla_X \left(\langle X, X \rangle - 1 \right) &= 0 \\ 2\widehat{\mathsf{L}} X - \lambda \left(2X \right) &= 0 \\ \implies \quad \widehat{\mathsf{L}} X - \lambda X &= 0 \\ \implies \quad \widehat{\mathsf{L}} X &= \lambda X \end{aligned} \tag{2.34}$$

which implies that X is an eigenvector of $\widehat{\mathsf{L}}$ with corresponding eigenvalue λ ($X \neq 0$ from the second equation of eq. (2.33)). Thus the two hypotheses are exactly equivalent when X is normalized. It is also worth remarking that the corresponding eigenvalue λ is the Lagrangian multiplier itself. □

Thus, to find the directions of greatest and least curvature of a surface at a point $u \in U$, we simply must calculate the Weingarten map and its eigenvectors. We refer to these directions as follows.

Definition 2.12 (Principal Curvatures and Principal Directions)**.** *The extremal values of normal curvature of a surface at a point $u \in U$ are referred to as* ***principal curvatures****. The corresponding directions at which normal curvature attains an extremal value are referred to as* ***principal directions****.*

Our final goal is to explicitly determine a (hopefully simplified) version of the Weingarten map in the case of a graph $f(u_1, u_2) = (u_1, u_2, h(u_1, u_2))$ and calculate the principal directions and curvatures in a simple example.

Theorem 2.6 (Relationship between Hessian and Weingarten Map of a Graph)**.** *Given the graph $f : U \to \mathbb{R}^3$ where $(x, y) \mapsto (x, y, h(x, y))$, the matrix representation of its Weingarten map*

is given by

$$\widehat{\mathsf{L}} = \mathrm{Hess}(h)\tilde{G}\,, \quad \text{where} \quad \tilde{G} := \frac{1}{\sqrt{1+h_x^2+h_y^2}}\begin{bmatrix} 1+h_y^2 & -h_x h_y \\ -h_x h_y & 1+h_x^2 \end{bmatrix} \tag{2.35}$$

In particular, given a point $u = (x, y) \in U \subset \mathbb{R}^2$ where $h_x \approx h_y \approx 0$, we have $\tilde{G} \approx \mathrm{Id}$, and thus $\widehat{\mathsf{L}} \approx \mathrm{Hess}(h)$.

Proof. First, we can (using chain rule) rewrite each component as in eq. (2.23):

$$h_{ij} = \langle \frac{\partial f}{\partial u_i}, -\frac{\partial \nu}{\partial u_j} \rangle = \langle \frac{\partial^2 f}{\partial u_i \partial u_j}, \nu \rangle$$

Now, given our particular surface f, we can calculate each of these components directly. We have:

$$\begin{gathered} f_x = (1,0,h_x), \quad f_y = (0,1,h_y) \\ f_{xx} = (0,0,h_{xx}), \quad f_{xy} = (0,0,h_{xy}) = f_{yx}, \quad f_{yy} = (0,0,h_{yy}) \end{gathered} \tag{2.36}$$

and we have the unit normal vector (Gauss map)

$$\nu(u_1,u_2) = \frac{\frac{\partial f}{\partial x} \times \frac{\partial f}{\partial y}}{\|\frac{\partial f}{\partial x} \times \frac{\partial f}{\partial y}\|} \tag{2.37}$$

$$= \frac{(1,0,h_x) \times (0,1,h_y)}{\|(1,0,h_x) \times (0,1,h_y)\|} \tag{2.38}$$

$$= \frac{\left(-h_x, -h_y, 1\right)}{\sqrt{h_x^2+h_y^2+1}} \tag{2.39}$$

We then calculate each h_{ij} as

$$\begin{aligned} h_{11} &= \langle \frac{\partial^2 f}{\partial x^2}, \nu \rangle = \frac{h_{xx}}{\sqrt{1+h_x^2+h_y^2}} \\ h_{12} &= \langle \frac{\partial^2 f}{\partial x \partial y}, \nu \rangle = \frac{h_{xy}}{\sqrt{1+h_x^2+h_y^2}} = h_{21} \\ h_{22} &= \langle \frac{\partial^2 f}{\partial y^2}, \nu \rangle = \frac{h_{yy}}{\sqrt{1+h_x^2+h_y^2}} \end{aligned} \tag{2.40}$$

and thus the first matrix in eq. (2.25) is given by

$$[h_{ij}] = \frac{1}{\sqrt{1+h_x^2+h_y^2}}\text{Hess}(h) \tag{2.41}$$

To calcuate the second, we use

$$\begin{aligned} g_{ij} &= \langle \frac{\partial f}{\partial u_i}, \frac{\partial f}{\partial u_j} \rangle \\ g_{11} &= \langle f_x, f_x \rangle = 1+h_x^2 \\ g_{12} &= \langle f_x, f_y \rangle = h_x h_y = g_{21} \\ g_{22} &= \langle f_y, f_y \rangle = 1+h_y^2 \end{aligned} \tag{2.42}$$

and thus

$$[g_{ij}]^{-1} = \begin{bmatrix} 1+h_x^2 & h_x h_y \\ h_x h_y & 1+h_y^2 \end{bmatrix}^{-1} = \begin{bmatrix} 1+h_y^2 & -h_x h_y \\ -h_x h_y & 1+h_x^2 \end{bmatrix} \tag{2.43}$$

Combining $[h_{ij}]$ and $[g_{ij}]^{-1}$ from eq. (2.43) and eq. (2.41) we arrive at eq. (2.35). □

Thus the matrix of the Weingarten map $\widehat{\mathsf{L}}$ is the Hessian matrix exactly at a critical point $u \in U$, where $\nabla h(u) = (h_x(u), h_y(u)) = 0$. Of course this implies that $\widehat{\mathsf{L}}$ and $\text{Hess}(h)$ have the same eigenvalues and eigenvectors at these points.

But this observation is more broadly useful than that, since if $\tilde{G}$ above is close to identity, then the eigenvalues and eigenvectors of $\widehat{\mathsf{L}}$ will be similarly close to the eigenvalues of the Hessian. We can rewrite $\tilde{G}$ from eq. (2.35) as identity plus a small matrix:

$$\tilde{G} = I + [\delta], \quad [\delta] := \begin{bmatrix} h_y^2 & -h_x h_y \\ -h_x h_y & h_x^2 \end{bmatrix} \tag{2.44}$$

We can then rewrite eq. (2.35) as

$$\widehat{\mathsf{L}} = \frac{1}{\sqrt{1+h_x^2+h_y^2}}\text{Hess}(h) + \frac{1}{\sqrt{1+h_x^2+h_y^2}}\text{Hess}(h)[\delta] \tag{2.45}$$

We can see that as h_x, h_y are close to zero, $[\delta]$ will be very close to the zero matrix (and the constant $\frac{1}{\sqrt{1+h_x^2+h_y^2}}$ will be very close to 1 as well), and we should not expect the addition of a "close to 0" matrix to have much effect on the eigenvectors or eigenvalues. This intuition is confirmed by a result from Wilkinson [31], which we state without rigorous proof.

Theorem 2.7. *If A, B are matrices such that $|A_{ij}| < 1, |B_{ij}| < 1$ (a condition that can be ignored with scaling) and λ is a simple eigenvalue of A, then given $\epsilon > 0$, there exists a simple eigenvalue $\tilde{\lambda}$ of the matrix $A + \epsilon B$ with $|\lambda - \tilde{\lambda}| = O(\epsilon)$. Similarly, if v is an eigenvector of A, then $\tilde{v}$ is an eigenvector of $A + \epsilon B$ with $|v - \tilde{v}| = O(\epsilon)$.*

The proof ultimately relies on a general result of analysis, that the zeros of a polynomial are continuous with respect to its coefficients. In this case, the polynomial in question is the characteristic polynomial $p(\lambda) = \det(\lambda I - A - \epsilon B)$, whose coefficients will scale with ϵ. Thus $\widehat{\mathsf{L}} \approx \text{Hess}(h)$ for any point where the gradient $\nabla h \approx 0$. We shall see that we're only concerned with regions where h_x, h_y is small anyway, and we do not expect

In the event that we do wish to rigorously compute the Weingarten map should want to be rigorously computed "without approximation"–that is, without concern for the magnitude of the gradient–we refer to [14] and survey papers mentioned therein.

To make the Weingarten map and its relationship to the Hessian more explicit, we will calculate the Weingarten map for a relatively simple graph.

The Weingarten map and Principal Curvatures of a Cylindrical Ridge

Let f be the graph given by

$$f : \mathbb{R}^2 \to \mathbb{R}^3 \text{ by } f(x,y) = (x,y,h(x,y)), \text{ with } h(x,y) = \begin{cases} \sqrt{r^2 - x^2} & -r \le x \le r \\ 0 & \text{else} \end{cases} \tag{2.46}$$

The graph is shown in fig. 2. We calculate the necessary partial derivatives of f as follows:

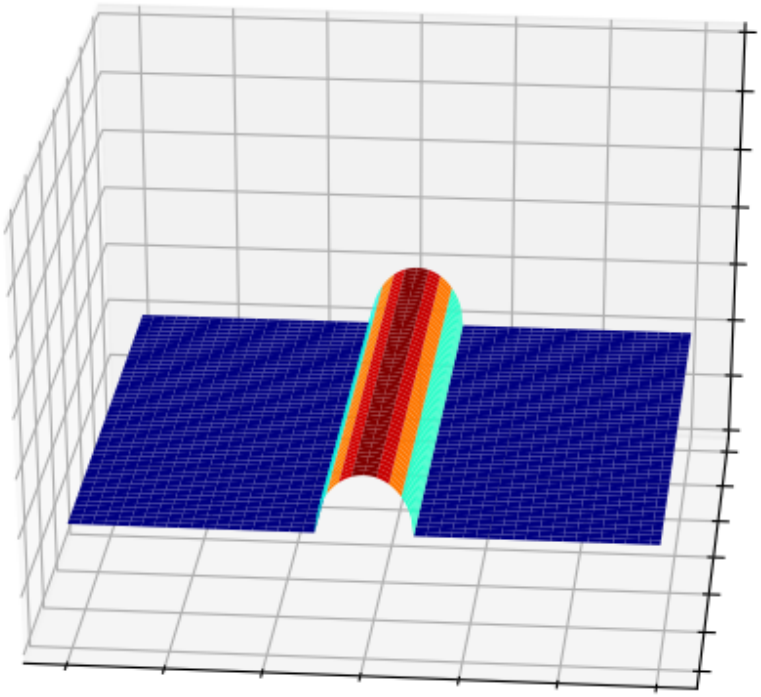


FIGURE 2: The graph of a cylindrical ridge of radius r

$$\frac{\partial f}{\partial x} = \left(1, 0, \frac{-x}{\sqrt{r^2 - x^2}}\right) \quad , \quad \frac{\partial^2 f}{\partial x^2} = \left(0, 0, \frac{-r^2}{\left(\sqrt{r^2 - x^2}\right)^3}\right) \tag{2.47}$$

$$\frac{\partial f}{\partial y} = (0, 1, 0) \quad , \quad \frac{\partial^2 f}{\partial y^2} = \frac{\partial^2 f}{\partial x \partial y} = 0 \tag{2.48}$$

The gauss map is given by

$$\nu(x, y) = \frac{\frac{\partial f}{\partial x} \times \frac{\partial f}{\partial y}}{\| \frac{\partial f}{\partial x} \times \frac{\partial f}{\partial y} \|} = \left(\frac{x}{r},\, 0,\, \frac{\sqrt{r^2 - x^2}}{r}\right) \tag{2.49}$$

$$\Longrightarrow \frac{\partial \nu}{\partial x} = \left(\frac{1}{r}\,,\, 0\,,\, \frac{-x}{r\sqrt{r^2 - x^2}}\right) \quad , \quad \frac{\partial \nu}{\partial y} = (0, 0, 0)\,. \tag{2.50}$$

We then calculate matrix elements of the Weingarten map's construction as given in eq. (2.41) and eq. (2.43) :

$$[h_{ij}] = \frac{1}{\sqrt{1+h_x^2+h_y^2}}\text{Hess}(h) = \frac{1}{\sqrt{1+\left(\frac{x^2}{r^2-x^2}\right)}}\begin{bmatrix}\frac{-r^2}{\sqrt{r^2-x^2}^3} & 0\\ 0 & 0\end{bmatrix} = \begin{bmatrix}\frac{-r}{r^2-x^2} & 0\\ 0 & 0\end{bmatrix} \tag{2.51}$$

$$[g_{ij}]^{-1} = \begin{bmatrix}\frac{r^2-x^2}{r^2} & 0\\ 0 & 1\end{bmatrix} \tag{2.52}$$

$$\implies \widehat{\mathsf{L}} = [h_{ij}][g_{ij}]^{-1} = \begin{bmatrix}\frac{-r}{r^2-x^2} & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}\frac{r^2-x^2}{r^2} & 0\\ 0 & 1\end{bmatrix} \tag{2.53}$$

$$= \begin{bmatrix}-\frac{1}{r} & 0\\ 0 & 0\end{bmatrix} \tag{2.54}$$

We see that $u_2 = (0,1)$ and $u_1 = (1,0)$ are eigenvectors for $\widehat{\mathsf{L}}$ with respective eigenvalues $\kappa_2 = -\frac{1}{r}, \kappa_1 = 0$. Given the theorem of Olinde Rodriguez suggests that u_2 points in the direction of maximum curvature of the surface, $-\frac{1}{r}$, which is predictably in the direction directly perpendicular to the trough, whereas the direction of least curvature is along the trough and is 0. The theorem of Meusnier theorem 2.2 suggests that the normal curvature $\kappa_2 = -\frac{1}{r}$ is reasonable– any curve on the trough perpendicular to the ridge should have the curvature of a circle (the negative simply indicates that we are on the "outside" of the surface). Finally, we note that at the ridge of the trough is exactly where $\nabla f = 0$, and the Weingarten map is exactly the Hessian matrix there.

Viewing the surface in $\mathbb{R}^3$, we define the Hessian $\text{Hess}(x,y)$ of the surface L at a point (x,y) on the surface as the matrix of its second partial derivatives:

$$\text{Hess}(x,y) = \begin{bmatrix}L_{xx}(x,y) & L_{xy}(x,y)\\ L_{yx}(x,y) & L_{yy}(x,y)\end{bmatrix} \tag{2.55}$$

At any point (x,y) we denote the two eigenpairs of $\text{Hess}(x,y)$ as

$$\text{Hess}(x,y)u_i = \kappa_i u_i\,, \quad i = 1,2 \tag{2.56}$$

where κ_i and u_i are known as the *principal curvatures* and *principal directions* of $L(x, y)$, respectively, and we label such that $|\kappa_2| \geq |\kappa_1|$. Notably, Hess$(x, y)$ is a real, symmetric matrix (since $L_{xy} = L_{yx}$ and L is a real function) and thus its eigenvalues are real and its eigenvectors are orthonormal to each other, as given by following basic result from linear algebra, [4]:

Lemma 2.8 (Principal Axis Theorem?)**.** *Let A be a real, symmetric matrix. The eigenvalues of A are real and its eigenvectors are orthonormal to each other.*

Proof. Let $x \neq 0$ so that $Ax = \lambda x$. Then

$$\begin{aligned} ||Ax||_2^2 = \langle Ax, Ax \rangle = (Ax)^* Ax \\ &= x^* A^* Ax = x^* A^T Ax = x * AAx \\ &= x^* A \lambda x = \lambda x^* Ax \\ &= \lambda x^* \lambda x = \lambda^2 x^* x = \lambda^2 ||x||_2^2 \end{aligned}$$

Upon rearrangement, we have $\lambda^2 = \frac{||Ax||_2^2}{||x||_2^2} \geq 0 \implies \lambda$ is real.

To prove that a set of orthonormalizable eigenvectors exists, let A be real, symmetric as above and consider the eigenpairs $Av_1 = \lambda_1 v_1$, $Av_2 = \lambda_2 v_2$ with $v_1, v_2 \neq 0$. [1]

In the case that $\lambda_1 \neq \lambda_2$, we have

$$\begin{aligned} (\lambda_1 - \lambda_2) v_1^T v_2 = \lambda_1 v_1^T v_2 - \lambda_2 v_1^T v_2 \\ &= (\lambda_1 v_1)^T v_2 - v_1^T (\lambda_2 v_2) \\ &= (Av_1)^T v_2 - v_1^T (Av_2) \\ &= v_1^T A^T v_2 - v_1^T A v_2 \\ &= v_1^T A v_2 - v_1^T A v_2 = 0 \end{aligned}$$

Since $\lambda_1 \neq \lambda_2$, we conclude that $v_1^T v_2 = 0$.

[1] To simplify notation, we simplify our argument to consider two explicit eigenvectors only, since we're only concerned with the 2×2 matrix Hess anyway.

In the case that $\lambda_1 = \lambda_2 =: \lambda$, we can define (as in Gram-Schmidt orthogonalization) $u = v_2 - \frac{v_1^T v_2}{v_1^T v_1} v_1$. This is an eigenvector for $\lambda = \lambda_2$, as

$$\begin{aligned} Au &= A\left(v_2 - \frac{v_1^T v_2}{v_1^T v_1} v_1\right) \\ &= Av_2 - \frac{v_1^T v_2}{v_1^T v_1} Av_1 \\ &= \lambda v_2 - \frac{v_1^T v_2}{v_1^T v_1} \lambda v_1 \\ &= \lambda\left(v_2 - \frac{v_1^T v_2}{v_1^T v_1} v_1\right) = \lambda u \end{aligned}$$

and is perpendicular to v_1, since

$$\begin{aligned} v_1^T u &= v_1^T\left(v_2 - \frac{v_1^T v_2}{v_1^T v_1} v_1\right) \\ &= v_1^T v_2 - \left(\frac{v_1^T v_2}{v_1^T v_1}\right) v_1^T v_1 \\ &= v_1^T v_2 - v_1^T v_2 (1) = 0. \end{aligned}$$

□

Thus we see that the two principal directions form an orthonormal frame at each point (x,y) within the continuous image $L(x, y)$.

We now seek to harness the ideas of this section to the task at hand: identifying curvilinear content within images.

The Frangi Filter: Uniscale

The Frangi filter, first described by Alejandro Frangi et al. in [10] is a widely used (cite) Hessian-based filter within image processing. Hessian-based filters make use of the logical "proximity" of the Hessian to notions of curvature of surfaces, as developed in section 2.2. Several such Hessian-based filters exist–see [28] and [24], as well as a comparison given in [27]. These filters use information about the principal curvatures,

approximated as eigenvalues of the Hessian) at each point in the image to identify regions of significant curvature within an image.

Frangi's filter was orignally developed for vascular segmentation in images such as MRIs and it excels in that context.

The procedure for a single scale in a 2D image is as follows: Let λ_1, λ_2 be the two eigenvalues of the Hessian of the image at point (x, y), ordered such that $|\lambda_1| \leq |\lambda_2|$, and define the Frangi vesselness measure as:

$$V_\sigma(x_0, y_0) = \begin{cases} 0 & \text{if} \quad \lambda_2 > 0 \\ \exp\left(-\frac{A^2}{2\beta^2}\right)\left(1 - \exp(-\frac{S^2}{2\gamma^2})\right) & \text{otherwise} \end{cases} \tag{2.57}$$

where

$$A := |\lambda_1/\lambda_2| \quad \text{and} \quad S := \sqrt{\lambda_1^2 + \lambda_2^2} \tag{2.58}$$

and β and γ are tuning parameters. Before we discuss appropriate values for β and γ, we first seek to highlight the significance of eq. (2.57), and in particular, the ratios defined in eq. (2.58). A and S are known as the anisotropy measure and structureness measure, respectively.

Anisotropy Measure

The anisotropy (or directionality) measure A is simply the ratio of magnitudes of λ_1 and λ_2. Since at a ridge point of a tubular structure, we should have $\lambda_1 \approx 0$ and $|\lambda_2| \gg |\lambda_1|$, a very small value of A would be present at a ridge of a tubular structure.

In fig. 3, this situation is demonstrated. Here, u_1, u_2 form the orthogonal set of Hessian eigenvectors with corresponding eigenvalues λ_1 and λ_2. At such a ridgelike structure, we could predict the largest change in curvature to be straight down the ridge (in the direction of u_2), and the direction of least curvature to be directly along the ridge (in the direction of u_1). $\lambda_1 \approx 0$ and λ_2 is large and negative Note that the length of these vectors in this picture is not meant tor represent their magnitudes, as u_2 should have a

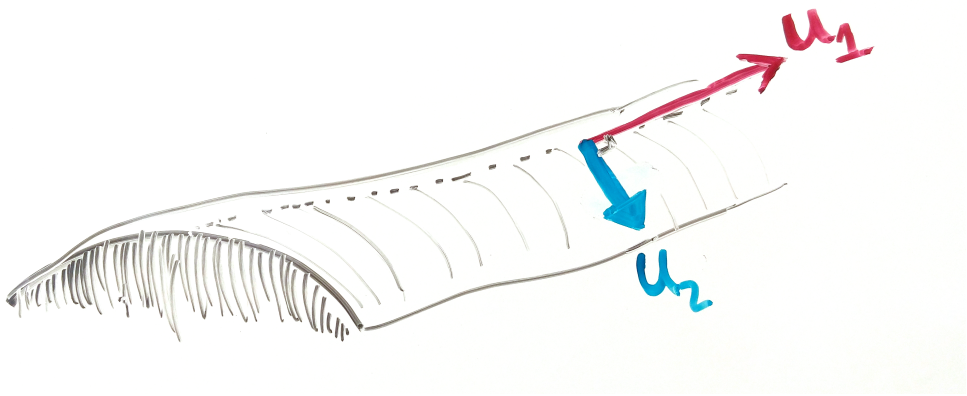


FIGURE 3: The principal eigenvectors at a ridge like structure

much larger relative magnitude by design!

Of course, if the the ridge is perfectly circular along its cross section (as was in section 2.2.4, it is of course apparent that λ_2 would be the same value at any place along the ridge (not just at its crest), and λ_1 would likewise be 0 at any such point. One could also imagine a similar situation in which the dropoff from crest to bottom gets increasing steep. In such a case, λ_2 as a function of x would in fact be largest nearest to the bottom. This thought experiment should dispel a naive misunderstanding of the power of a Frangi filter: a high anisotropy measure (and a large structureness measure) will not in general identify the crests of a ridge-like structure–it only will highlight that such a pixel is on a ridge-like structure at all. Thus, the anisotropy measure will not necessarily be at a maximum at the crest of the ridge.

Similiarly, the vessel we we wish to identify can not be reasonably expected to behave as perfectly as our toy example. There will likely be small aberrations in a ridgelike structure, such as small divots or depressions in an overall ridge-like structure. Of importance in our data set later (section 4.1), there will be points where we seem to ”lose” our ridgelike structure, but this is simply due to an error in the sample.

Importantly, this formulation does not require λ_1 to be approximately zero, just that the curvature in the downward direction is much more significant.

Also the crest could be really flat ("hangar shaped"), in which case both are around zero. At the crest of the ridge, we would actually expect both u_1 and u_2 to be around 0, whereas a point somewhere between the crest and the "foot" of the ridge to contain the maximum u_2.

We will fix some of these issues by casting this as a multiscale problem in section 2.5.

Two other ideas that could fix some other discrepancies mentioned above is to identify these ridges on their own, or also where the 'feet are'. We will discuss these ideas in section 6.2.

Structureness measure

There is another concern with using the pure ratio $S := |\lambda_1/\lambda_2|$ as an identifying feature of ridgelike structures apart from the ones listed above. We could still have $|\lambda_2| \gg |\lambda_1|$ in a relative sense, but still have $\lambda_2 \approx 0$. As a rather extreme example, we should certainly wish to differentiate a point on the surface where $\lambda_2 \approx 10^{-5}$ and $\lambda_1 \approx 10^{-10}$ from another point where $\lambda_2 \approx 10000$ and $\lambda_2 = 0.1$.

A natural fix to differentiate these points is to introduce a "structureness" measure to insure that there is in fact significant curvilinear activity at the point in question. Frangi used $S := \sqrt{(\lambda_1)^2 + (\lambda_2)^2}$, which is in fact the 2-norm of the Hessian matrix. Thus the Frangi filter should also prefer areas of great curvilinear content in the image first of all.

The Frangi vesselness measure

Our goal then is to attach a numerical measure to each pixel in the image (at a particular scale σ) that is large when the anisotropy measure A and the structureness measure S is sufficiently large.

The form Frangi arrived at in eq. (2.57) in which a factor of exp{...} and $(1-\exp\{\})$ are multiplied together are simply to ensure that the final vesselness measure V is largest when A is small and S is large enough, with rapidly decay in other situations.

Frangi further strengthened the filter by adding an additional case to in eq. (2.57), ensuring that λ_2 is not positive. If we are indeed at a curvilinear ridge, we need the second derivative of the surface in the maximal direction to be negative, which hasn't been accounted for as yet in our formulation of A and S – we wish (for our purposes) to only indentify when we are finding crests. A will still be small and S will still be large however if we identify a "trough".

The only perceivable difference is that the maximum normal curvature will be positive–we are at a local minimum in the direction of u_2. In situations where we wish to only identify ridges (as is the case here) we simply exclude any points where there is not a negative curvature in the maximal direction.

The Frangi vesselness filter: Choosing parameters β and γ

The parameters β and γ are meant to scale so that the peaks of $\exp\{\cdots\}$ and $1-\exp\{\cdots\}$ coincide enough to be statistically significant but rapidly decay in areas not associated with curvilinear structure. What values of these parameters are appropriate is ultimately dependent on the context of the problem.

Frangi suggested for γ that half of the Frobenius norm of the Hessian matrix is appropriate, simply because the minimum value of S is zero, and its maximum value is approx the 2 norm of the Hessian. For β Frangi chose an innocuous intermediate point, $\beta = 1/2$ (and thus $2\beta^2 = 1/2$). As we will show later, choosing the structureness parameter γ is rather important for the context especially if the background (non-ridgelike structure) is significant and noisy. β should be strengthened/relaxed depending on how "flat" the ridgelike structure is. If there is a lot of gain then β should be smaller. If this is not the case, a stronger filter can be created by requiring A to be much smaller.

We now take a quick tangent from our description of the Frangi filter to develop and justify our "multiscale" approach.

Linear Scale Space Theory

There is obviously a major disconnect in the ideas presented above. Although the ideas presented above require differentiation of continuous surfaces, our image is in fact a discrete pixel. That is, our previous discussions have been in terms of an image as the continuous surface in definition 2.2, rather than the more realistic discrete pixel matrix as in definition 2.1. The present section seeks to address this disconnect. In particular, we seek to mitigate the bias of our limited sampling of the "true" 3D surface. Our main goal is to counter against some of the bias of our particular sampling. In particular, we wish to not over-represent structures that are clear at our resolution without giving appropriate weight to larger structures as well. Koenderink [16] argued that "any image can be embedded in a one-parameter family of derived images (with resolution as the parameter) in essentially only one unique way" given a few of the so-called scale space axioms. He (and others) showed that a small set of intuitive axioms imply require that any such family of images must satisfy the heat equation

$$\Delta K(x,y,\sigma) = K_\sigma(x,y,\sigma) \text{ for } \sigma \geq 0 \text{ such that } K(x,y,0) = u_0(x,y). \tag{2.59}$$

where $K : \mathbb{R}^3 \to \mathbb{R}$ and $u_0 : \mathbb{R}^2 \to \mathbb{R}$: is the original image (viewed as a continuous surface) and σ is a resolution parameter. Much work has been done to formalize this approach [29]. There is a long list of desired properties–we will to try to identify a minimal subset of axioms and show that other desired properties follow.

Axioms

To make matters manageable, we require the one-parameter family of scaled images to be generated by an operation on the original image:

$$\{\, K(x,y;\sigma) = T_\sigma u_0 \mid \sigma \geq 0\ ,\ K(x,y,;0) = u_0 \,\}$$

The following axioms are then requirements on what sort of operation T_σ should be.

Axiom 2.1 (Linear-shift and Rotational Invariance)**.** *Linear-shift (or translation) invariance means that no position in the original signal is favored. This is intuitive, as our operation should apply to any image fairly, regardless of where content is found in the image. Similarly, there should be not be favoritism toward any particular orientation of content within the image.*

Axiom 2.2 (Continuity of Scale Parameter)**.** *There is no reason for the scale parameter to be discrete; we may alter the resolution with whatever precision we desire. That is, we take the resolution parameter σ to be a nonzero real number (as opposed to an integer). Moreover, we require that the operator behaves continuously with respect to the scale parameter.*

What happens as $\sigma \downarrow 0$ is not immediately clear though. An argument from functional analysis (see [12]) implies that there is a so-called "infinitesimal generator" A which is a limit case of our desired operator T; that is

$$Au_0 = \lim_{\sigma \downarrow 0} \frac{T_\sigma u_0 - u_0}{\sigma} \tag{2.60}$$

and moreover that there is a resultant differential equation concerning the derivative of the family and A:

$$\partial_\sigma K(x,y;\sigma) = \lim_{\sigma \downarrow 0} \frac{K(\cdot;\sigma+h) - K(\cdot;\sigma)}{h} = A(T_\sigma u) = A(K(\cdot,\sigma)) \tag{2.61}$$

We shall return to this idea later and more concretely describe A once we actually characterize the generating operator T_σ.

Axiom 2.3 (Semigroup property)**.** *The semigroup property is simply that transforming the original image by some resolution σ should have the same overall effect of two successive transformations σ_1 and σ_2, i.e.*

$$T_\sigma u = T_{\sigma_1+\sigma_2} u \tag{2.62}$$

Axiom 2.4 (Causality Conditon)**.** *The following requirement has great implication, and is also very successful in encoding our intuitive sense of "resolution". The causality condition is the one that, as resolution decreases, no finer detail is introduced into the image. That is, as the scale increases, there will be no creation of local extrema that did not exist at a smaller scale.*

In other words, if $K(x_0, y_0; \sigma_0)$ is a local maximum (at the point (x_0, y_0), at this fixed σ_0) i.e. then an increase in scale can only weaken this peak, i.e.

$$\begin{cases} \nabla K(x_0, y_0; \sigma_0) = 0 \\ \Delta K(x_0, y_0; \sigma_0) < 0 \end{cases} \implies K(x_0, y_0; \sigma_1) \leq K(x_0, y_0; \sigma_0) \; \forall \; \sigma_1 \geq \sigma_0 \tag{2.63}$$

Similarly, if $K(x_0, y_0; \sigma_0)$ is a local minimum (with respect to space), then an increase in scale cannot make such a valley more profound, i.e.

$$\begin{cases} \nabla K(x_0, y_0; \sigma_0) = 0 \\ \Delta K(x_0, y_0; \sigma_0) > 0 \end{cases} \implies K(x_0, y_0; \sigma_1) \geq K(x_0, y_0; \sigma_0) \; \forall \; \sigma_1 \geq \sigma_0 \tag{2.64}$$

This implies that no image feature is sharpened by an decrease and resolution–the only result is a monotonic blurring of the image as scale parameter σ tends to infinity.

Uniqueness of the Gaussian Kernel

The above requirements are actually sufficient in proving not only that the operator T_σ is a convolution, but that the heat equation described in eq. (2.59) must hold. This has been shown in various ways, both by Koenderink [16], Babaud [3], as well as Lindeberg in [29]. In fact, it is shown that the Gaussian is the unique convolution kernel that works.

To this, show that:

- a kernel satisfying the above axioms must satisfy the heat equation
- the gaussian kernel satisfies that.
- gaussian kernel is the only kernel that works.

That is,

$$K(x,y;\sigma) = T_\sigma u_0 = G_\sigma \star u_0 \quad \text{where} \quad G_\sigma := \frac{1}{2\pi\sigma^2} e^{\left(-|x|^2/(2\sigma^2)\right)} \tag{2.65}$$

We can show that this solution solves the heat equation. Given u_0 as a continuous image (unscaled), we construct PDE with this as a boundary condition.

$$u : \mathbb{R}^2 \supset \Omega \to \mathbb{R} \text{ with } u(\boldsymbol{x},t) : \begin{cases} \frac{\partial u}{\partial t}(\boldsymbol{x},t) = \Delta u(\boldsymbol{x},t) \quad , \ t \geq 0 \\ u(\boldsymbol{x},0) = u_0(\boldsymbol{x}) \end{cases} \tag{2.66}$$

We show that

$$u(\boldsymbol{x},t) = \left(G_{\sqrt{2t}} \star u_0\right)(\boldsymbol{x}) \tag{2.67}$$

solves (the above tagged equation), where

s

First, we need a quick lemma regarding differentiation a continuous convolution.

Lemma 2.9. *Derivative of a convolution is the way that it is (obviously rewrite this).*

Proof. For a single variable,

$$\frac{\partial}{\partial\alpha}\left[f(\alpha) \star g(\alpha)\right] = \frac{\partial}{\partial\alpha}\left[\int f(t)g(\alpha - t)dt\right] \tag{2.68}$$

$$= \int f(t)\frac{\partial}{\partial\alpha}\left[g(\alpha - t)\right]dt \tag{2.69}$$

$$= \int f(t)\left(\frac{\partial g}{\partial\alpha}\right)g(\alpha - t)dt \tag{2.70}$$

$$= f(\alpha) \star g'(\alpha) \tag{2.71}$$

By symmetry of convolution we can also conclude

$$\frac{\partial}{\partial \alpha}[f(\alpha) \star g(\alpha)] = f'(\alpha) \star g(\alpha)$$

If f and g are twice differentiable, we can compound this result to show a similar statement holds for second derivatives, and then, given the additivity of convolution, we may conclude

$$\Delta(f \star g) = \Delta(f) \star g = f \star \Delta(g) \tag{2.72}$$

□

Theorem 2.10. *$u(\boldsymbol{x}, t) = \left(G_{\sqrt{2t}} \star u_0\right)(\boldsymbol{x})$ solves the heat equation.*

Proof. We focus on the particular kernel

$$G_{\sqrt{2t}} = \frac{1}{4\pi t} e^{(-|x|^2/(4t))}$$

Then

$$\begin{aligned}
\frac{\partial u}{\partial t}(\boldsymbol{x}, t) &= \frac{\partial}{\partial t}\left(G_{\sqrt{2t}}(\boldsymbol{x}, t) \star u_0(\boldsymbol{x})\right) && (2.73)\\
&= \frac{\partial}{\partial t}\left(G_{\sqrt{2t}}(\boldsymbol{x}, t)\right) \star u_0(\boldsymbol{x}) && (2.74)\\
&= \frac{\partial}{\partial t}\left(\frac{1}{4\pi t} e^{(-|x|^2/(4t))}\right) \star u_0(\boldsymbol{x}) && (2.75)\\
&= \left[-\frac{1}{4\pi t^2} e^{(-|x|^2/(4t))} + \frac{1}{4\pi t}\left(\frac{-|x|^2}{4t^2}\right) e^{-|x|^2/(4t)}\right] \star u_0(\boldsymbol{x}) && (2.76)\\
&= -\frac{1}{4t^2}\left(e^{(-|x|^2/(4t))} + |\boldsymbol{x}|^2 G_{\sqrt{2t}}(\boldsymbol{x}, t)\right) \star u_0(\boldsymbol{x}) && (2.77)
\end{aligned}$$

and from the previous lemma,

$$\Delta u(\boldsymbol{x}, t) = \Delta\left(G_{\sqrt{2t}} \star u_0(\boldsymbol{x})\right) = \Delta\left(G_{\sqrt{2t}}\right) \star u_0(\boldsymbol{x})$$

We explicitly calculate the Laplacian of $G_\sigma(x, y) = A \exp(-\frac{x^2+y^2}{2\sigma^2})$ as follows:

$$\frac{\partial}{\partial x}G_\sigma(x,y) = A\left(\frac{-2x}{2\sigma^2}\right)\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

$$\begin{aligned}\implies \frac{\partial^2}{\partial^2 x}G_\sigma(x,y) &= A\cdot\frac{\partial}{\partial x}\left[-\frac{x}{\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)\right]\\ &= A\left[-\frac{1}{\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)+\frac{x}{\sigma^2}\cdot\frac{2x}{2\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)\right]\\ &= A\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)\left[-\frac{1}{\sigma^2}+\frac{x^2}{\sigma^4}\right]\\ &= \frac{1}{\sigma^2}G_\sigma(x,y)\left[\frac{x^2}{\sigma^2}-1\right]\end{aligned}$$

By symmetry of argument we also may conclude

$$\frac{\partial^2}{\partial y^2}G_\sigma(x,y) = \frac{1}{\sigma^2}G_\sigma(x,y)\left[\frac{y^2}{\sigma^2}-1\right]$$

and so

$$\Delta G_\sigma(x,y) = \frac{\partial^2}{\partial x^2}(G_\sigma)+\frac{\partial^2}{\partial y^2}(G_\sigma) = \frac{1}{\sigma^2}G_\sigma(x,y)\left[\frac{x^2+y^2}{\sigma^2}-2\right] \tag{2.78}$$

Then, given lemma 2.9, we conclude

$$\Delta\left[G_\sigma(x,y)\star u_0(x,y)\right] = \left(\frac{1}{\sigma^2}G_\sigma(x,y)\left[\frac{x^2+y^2}{\sigma^2}-2\right]\right)\star u_0(x,y) \tag{2.79}$$

For particular choices of $\sigma(t) = \sqrt{2t}$ and $A = \frac{1}{4\pi t}$, we see

$$\Delta\left[G_{\sqrt{2t}}(x,y)\star u_0(x,y)\right] = \left(\frac{1}{2t}G_{\sqrt{2t}}(x,y)\left[\frac{x^2+y^2}{2t}-2\right]\right)\star u_0(x,y) \tag{2.80}$$

$$= \left(G_{\sqrt{2t}}(x,y)\left[\frac{x^2+y^2}{4t^2}-\frac{1}{t}\right]\right)\star u_0(x,y) \tag{2.81}$$

We then calculate the time derivative, using our particular choice of $\sigma(t) = \sqrt{2t}$ and $A = \frac{1}{4\pi t}$ as:

$$\frac{\partial}{\partial t}\Big[G_{\sigma(t)}(x,y) \star u_0(x,y)\Big] = \frac{\partial}{\partial t}\Big[G_{\sigma(t)}(x,y)\Big] \star u_0(x,y) \tag{2.82}$$

$$= \frac{\partial}{\partial t}\Big[G_{\sqrt{2t}}(x,y)\Big] \star u_0(x,y) \tag{2.83}$$

$$= \frac{\partial}{\partial t}\left[\frac{1}{4\pi t}\exp\left(-\frac{x^2+y^2}{4t}\right)\right] \star u_0(x,y) \tag{2.84}$$

$$= \left[-\frac{1}{4\pi t^2}\exp\left(-\frac{x^2+y^2}{4t}\right) + \frac{1}{4\pi t}\left(\frac{x^2+y^2}{4t^2}\exp\left(-\frac{x^2+y^2}{4t}\right)\right)\right] \star u_0(x,y) \tag{2.85}$$

$$= \left(G_{\sqrt{2t}}(x,y)\left[\frac{x^2+y^2}{4t^2} - \frac{1}{t}\right]\right) \star u_0(x,y) \tag{2.86}$$

Combining these results, we find that

$$\frac{\partial}{\partial t}\Big[G_{\sqrt{2t}} \star u_0\Big] = \Delta\Big[G_{\sqrt{2t}} \star u_0\Big] \tag{2.87}$$

as desired. □

Scale Spaces over Discrete Structures

The above developments from scale space axioms have (since their first appearance) been recast in terms of discrete structures (rather than continuous surfaces) as in [21]. However, we've chosen to present the above in their original continuous surface for clarity of argument. The discrete case is not much different– we still have the same axioms, and it can be shown that the family of scaled images must simply satisfy a discrete version of the However, viewing our actual image definition 2.1 as a sample of a continuous surface definition 2.2, we might naïvely expect our convolution by the Gaussian to "commute" with our supposed sampling of the continuous signal, or even that we could simply convolve our discrete signal with a discretely sampled Gaussian kernel. The latter in fact, seems to be an often implemented interpretation of scale space theory.

To be clear, the "sampled" 1D Gaussian Kernel we have in mind might be given by:

Definition 2.13 (Sampled Gaussian Kernel and Generated Family)**.**

$$g(n;\sigma) = \frac{1}{2\pi\sigma} e^{-n^2/2\sigma}, \quad -\infty < n < \infty$$

and the resulting (1D) convolution would be given by

$$K(x,\sigma) = \sum_{n=-\infty}^{\infty} g(n;\sigma) f(x-n) \quad \text{for} \quad x \in \mathbb{Z},\ \sigma > 0$$

The reality of the matter is that a discretely sampled Gaussian is not an appropriate kernel for creating discrete scale space. In [21] and in particular [20], Lindeberg demonstrated that the sampled Gaussian kernel violates not only semigroup property (axiom 2.3), but–much less forgivably–the causality property (axiom 2.3). There is absolutely no guarantee that convolution with a sampled Gaussian kernel will not create "spurious" structures as resolution increases.

Fortunately, Lindeberg was immediately able to remedy this by providing a discrete analogue of the Gaussian kernel, which does satisfy axiom 2.4 and axiom 2.3:

Definition 2.14 (Discrete Gaussian Kernel)**.** *The discrete Gaussian kernel, which can be shown to be a suitable generator for scale space, is given by*

$$T(n;\sigma) = e^{-\alpha\sigma} I_n(\alpha\sigma), \quad I_n(\sigma) = I_{-n}(\sigma) = (-1)^n J_n(i\sigma) \quad n \geq 0, \sigma, \alpha > 0 \tag{2.88}$$

where I_n are the modified Bessel functions of integer order based on the ordinary Bessel functions J_n, i.e.

$$I_n(x) = \sum_{m=0}^{\infty} \frac{1}{m!(m+n)!} \left(\frac{x}{2}\right)^{2m+n}, \quad n \geq 0$$

where we have taken the liberty of simplifying the typical definition [1] (which involves the gamma function), since we only desire Bessel functions of integer order. The

parameter α above is simply an optional scaling parameter which is simply set to 1 hereforth.

The derived family of 1D signals is then given by

$$K(x,\sigma) = \sum_{n=-\infty}^{\infty} T(n;t) f(x-n) \quad \text{for} \quad x \in \mathbb{Z},\ t > 0 \tag{2.89}$$

The compatibility of scale space theory and derivatives on discrete structures and extension to two dimensions was also demonstrated by Lindeberg in [22] and [23]. In particular, we may take derivatives of the convolutions of our discrete images using, say, a central difference. Lastly, the 2D version of the family given in eq. (2.89) can be obtained by independent convolution of its dimensions (i.e. it is separable). We will make these ideas explicit in chapter 3 and the Appendix.

With the ideas of scale established, we may return to our discussion of the Frangi filter.

The Frangi Filter: A multiscale approach

Our ideas of scale developed in the previous section imply that, if the ridgelike structures we wish to detect are more prominent at different scales, then a multiscale approach is the natural one. Considering our developments in section 2.3, we wish to probe at multiple scales regions that would receive a high vesselness score at any range, and consider them all together. Frangi [10] approached this problem by simply aggregating vesselness measure over all scales:

$$V(x_0, y_0) = \max_{\sigma \in \Sigma} V_\sigma(x_0, y_0) \tag{2.90}$$

where $\Sigma := \{\sigma_0, \sigma_1, \cdots, \sigma_N\}$ is a range of parameters at which to probe. These should be chosen to be representative enough of all scales where meaningful content is expected to be found.

Thresholding

After this procedure, we are left with a matrix with as many samples/pixels as the original image, all with a vesselness measure between 0 and 1 for each pixel in the image:

$$\mathsf{V}_\Sigma := [V(x,y)]_{\substack{0\le x<M \\ 0\le y<N}} \tag{2.91}$$

Notably, Frangi [10] refrained from explicitly interpreting the probablility assigned by eq. (2.90); that is–whether a particular point (x, y) in the image definitely a vessel or not. Instead, he cautioned that the result should not be used as a segmentation method alone, and that the size of the vasculature cannot be determined rigorously from the filter alone.

However, for the purposes of obtaining an intermediate result, we wish to be final about the whole matter and ultimately say whether or not a pixel does in fact corresponds to a curvilinear structure. A straightforward enough approach is to simply threshold at some fixed value. The resulting matrix can be given in terms of either eq. (2.90) or eq. (2.91)

$$V_{\Sigma,\alpha}(x,y) = \begin{cases} 1 & \text{if} \quad V(x,y) \ge \alpha \\ 0 & \text{else} \end{cases}, \quad \alpha > 0 \text{ for } \alpha \text{ fixed.} \tag{2.92}$$

We will discuss alternatives methods of aggregating results from our multiscale method, as well as optimal values for parameters and scales in chapter 3. As a final note, we admit that any future extensions of this work (as will be discussed in chapter 6) should not hold too much stock in this thresholded result, and analyzing the raw vesselness score eq. (2.91), or even the un-merged scale-wise scores, would be far more rewarding.

All that remains to describe mathematically is how to actually calculate the derivatives of our images and deal with the ultimately discrete nature of our samples.

Calculating the 2D Hessian

According to section 2.4.3, we may calculate derivatives of our structure by calculating a gradient on our convolved image. Our method of calculating the gradient of a matrix uses a second-order accurate central difference, as in [9]. Specific implementation will be discussed in chapter 3.

We note in passing that we may take the derivative of the Gaussian kernel and then convolve it, and the effect will be the same as if we had taken the derivative subsequently [11]. This could offer some computational speedup if we wish to run this procedure on many samples and fixed scale sizes, although we have implemented our scale spaces in the conventional way, as discussed in chapter 3.

Convolution Speedup via FFT

In practice, the convolutions described above are very slow for large scales (σ), as the size of the kernel is very large. Instead, we will perform a fast Fourier transform, which requires only $\mathcal{O}\left(N \cdot \log_2 N\right)$ operations for a one dimension signal of length N, as compared to the N^2 operations required of a conventional discrete Fourier transform [11]. We will briefly outline the theory of Fourier transforms.

Fourier Transform of a continuous 1D signal .

A periodic signal (real valued function) $f(t)$ of period T can be expanded in an infinite basis as follows:

$$f(t) = \sum_{-\infty}^{\infty} c_n e^{i\frac{2\pi n}{T}t}, \quad c_n = \frac{1}{T}\int_{-T/2}^{T/2} f(t)e^{-i\frac{2\pi n}{T}t}dt \tag{2.93}$$

The Fourier transform of a 1D continuous function is defined by

$$F(\mu) := \mathscr{F}\{f(t)\} = \int_{-\infty}^{\infty} f(t)e^{i2\pi\mu}dt \tag{2.94}$$

An inverse transform will then recover our original signal:

$$f(t) = \mathscr{F}^{-1}\{F(\mu)\} = \int_{-\infty}^{\infty} F(\mu)e^{i2\pi\mu t}dt \tag{2.95}$$

Together, eq. (2.94) and eq. (2.95) are referred to as the *Fourier transform pair* of the signal $f(t)$.

Fourier Transform of a Discrete 1D signal .

We wish to develop the Fourier transform pair for a discrete signal., following [11]. We frame the situation as follows: A continuous function $f(t)$ is represented as the sampled function $\tilde{f}(t)$ by multiplying it by a sampling (or impulse) function, an infinite series of discrete impulses with equal spacing ΔT:

$$s_{\Delta T}(t) := \sum_{n=-\infty}^{\infty} \delta[t - n\Delta T] \,, \quad \delta[t] = \begin{cases} 1 \,, & t = 0 \\ 0 \,, & t \neq 0 \end{cases} \tag{2.96}$$

where $\delta[t]$ is the discrete unit impulse.

The discrete sample $f(t)$ is then constructed from $f(t)$ by

$$\tilde{f}(t) = f(t) s_{\Delta T}(t) \tag{2.97}$$

From this we can calculate $\tilde{F}(t)$. Given the discrete signal $\tilde{f}$, we construct the transform $\tilde{F}(\mu) = \mathscr{F}\{\tilde{f}(t)\}$. by expanding $\tilde{f}$ in the same infinite basis as the continuous case.

$$\tilde{F}(\mu) = \sum_{n=-\infty}^{\infty} f_n e^{-i2\pi\mu n\Delta T} \,, \quad f_n = \tilde{f}(n) = f(n\Delta T) \tag{2.98}$$

The transform is a continuous function with period $1/\Delta T$.

2D DFT Convolution Theorem .

Theorem 2.11 (2D DFT Convolution Theorem)**.** *Given two discrete functions are sequences with the same length.* $f(x, y)$ *and* $h(x, y)$ *for integers* $0 < x < M$ *and* $0 < y < N$*, we can take the*

discrete fourier transform (DFT) of each:

$$F(u,v) := \mathcal{D}\{f(x,y)\} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-2\pi i\left(\frac{ux}{M}+\frac{vy}{N}\right)} \tag{2.99}$$

$$H(u,v) := \mathcal{D}\{h(x,y)\} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} h(x,y)e^{-2\pi i\left(\frac{ux}{M}+\frac{vy}{N}\right)} \tag{2.100}$$

and given the convolution of the two functions

$$(f \star h)(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)h(x-m,y-n) \tag{2.101}$$

then $(f \star h)(x,y)$ *and* $MN \cdot F(u,v)H(u,v)$ *are transform pairs, i.e.*

$$(f \star h)(x,y) = \mathcal{D}^{-1}\{MN \cdot F(u,v)H(u,v)\} \tag{2.102}$$

The proof follows from the definition of convolution, substituting in the inverse-DFT of f and h, and then rearrangement of finite sums.

Proof.

$$(f \star h)(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)h(x-m,y-n) \tag{2.103}$$

$$= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left(\sum_{p=0}^{M-1}\sum_{q=0}^{N-1} F(p,q)e^{2\pi i\left(\frac{mp}{M}+\frac{nq}{N}\right)}\right)\left(\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} H(u,v)e^{2\pi i\left(\frac{u(x-m)}{M}+\frac{v(y-n)}{N}\right)}\right) \tag{2.104}$$

$$= \left(\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} H(u,v)e^{2\pi i\left(\frac{ux}{M}+\frac{vy}{N}\right)}\right)\left(\sum_{p=0}^{M-1}\sum_{q=0}^{N-1} F(p,q)\left(\sum_{m=0}^{M-1} e^{2\pi i\left(\frac{m(p-u)}{M}\right)}\right)\left(\sum_{n=0}^{N-1} e^{2\pi i\left(\frac{n(q-v)}{N}\right)}\right)\right) \tag{2.105}$$

$$= \left(\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} H(u,v)e^{2\pi i\left(\frac{ux}{M}+\frac{vy}{N}\right)}\right)\left(\sum_{p=0}^{M-1}\sum_{q=0}^{N-1} F(p,q)\left(M\cdot\hat{\delta}_M(p-u)\right)\left(N\cdot\hat{\delta}_M(q-v)\right)\right) \tag{2.106}$$

$$= \left(\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} H(u,v)e^{2\pi i\left(\frac{ux}{M}+\frac{vy}{N}\right)}\right)\cdot MNF(u,v) \tag{2.107}$$

$$= MN\cdot\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)H(u,v)e^{2\pi i\left(\frac{ux}{M}+\frac{vy}{N}\right)} \tag{2.108}$$

$$= MN\cdot\mathcal{D}^{-1}\{FH\} \tag{2.109}$$

where

$$\hat{\delta}_N(k) = \begin{cases} 1 & \text{when } k = 0 \mod N \\ 0 & \text{else} \end{cases} \tag{2.110}$$

□

Above, we make use of the following lemma

Lemma 2.12. *Let j and k be integers and let N be a positive integer. Then*

$$\sum_{n=0}^{N-1} e^{2\pi i\left(\frac{n(j-k)}{N}\right)} = N\cdot\hat{\delta}_N(j-k) \tag{2.111}$$

Proof. Consider the complex number $e^{2\pi i(j-k)/N}$. Note first that this is an N-th root of unity, since

$$\left(e^{2\pi i(j-k)/N}\right)^N = e^{2\pi i(j-k)} = \left(e^{2\pi i}\right)^{(j-k)} = 1^{(j-k)} = 1$$

In other words, $e^{2\pi in(j-k)/N}$ is a root of $z^N - 1 = 0$, which we can factor as

$$z^N - 1 \;=\; (z-1)\Big(z^{n-1} + \cdots + z + 1\Big) \;=\; (z-1)\sum_{n=0}^{N-1} z^n. \tag{2.112}$$

thus giving us

$$0 = \Big(e^{2\pi i(j-k)/N} - 1\Big)\sum_{n=0}^{N-1} e^{2\pi in(j-k)/N} \tag{2.113}$$

To prove the claim in eq. (2.111), we consider two cases: First, if $j-k$ is a multiple of N, we of course have $e^{2\pi in(j-k)/N} = \Big(e^{2\pi i}\Big)^{n(j-k)/N} = 1$ and thus the left side of eq. (2.111) reduces to

$$\sum_{n=0}^{N-1} \Big(e^{2\pi i}\Big)^{n(j-k)/N} = \sum_{n=0}^{N-1} (1) = N$$

.

In the case that $j-k$ is *not* a multiple of N, we refer to eq. (2.113). The first factor is not zero since, $\Big(e^{2\pi i(j-k)/N}\Big) \neq 1$ (simply since $(j-k)/N$ is not an integer), and thus it must be that the second factor is 0:

$$\sum_{n=0}^{N-1} \Big(e^{2\pi i(j-k)/N}\Big)^n = 0$$

We can combine these two cases by invoking the definition of eq. (2.110), giving us the result. □

FFT

As noted, the above result applies to the Discrete Fourier Transform. We actually achieve a convolution speedup using a Fast Fourier Transform (FFT) instead. We follow the developments of [11]. For clarity, we present the following theorems which allow a framework to calculate a 2D Fourier transforms quickly.

First, a 2D DFT may actually be calculated via two successive 1D DFTs, which can be seen through a basic rearrangement, as follows:

$$F(\mu,\nu) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-i2\pi(\mu x/M+\nu y/N)} \tag{2.114}$$

$$= \sum_{x=0}^{M-1} e^{-i2\pi\mu x/M}\left[\sum_{y=0}^{N-1} f(x,y)e^{-i2\pi\nu y/N}\right] \tag{2.115}$$

$$= \sum_{x=0}^{M-1} e^{-i2\pi\mu x/M}\,\mathscr{F}_x\{f(x,y)\} \tag{2.116}$$

$$= \mathscr{F}_y\{\mathscr{F}_x\{f(x,y)\}\} \tag{2.117}$$

where $\mathscr{F}_{x'}$ refers to the 1D discrete Fourier transform of the function with respect to the variable x' only.

Thus, to calculate the fourier transform $F(u,v)$ at the point u,v requires the computation of the transform of length N for each iterated point $x \in 0,\cdots,M-1$. Thus there are MN complex multiplications and $(M-1)(N-1)$ complex additions in this sequence required for each point u,v that needs to be calculated. Overall, for all points that need to be calculated, the total order of calculations is on the order of $(MN)^2$ We'll also mention that the values of $e^{-i2\pi m/n}$ can be provided by a lookup table rather than ad-hoc calculation.

We now show that a considerable speedup can be achieved through elimination of redundant calculations. In particular, we wish to show that the calculation of a 1D DFT of signal length $M = 2^n, n \in \mathbb{Z}_+$ can be reduced to calculating two half-length transforms and an additional $M/2 = 2^{n-1}$ calculations.

To "simplify" our notation we will use a new notation for the Fourier kernels/basis functions. Let the 1D Fourier transform be given by

$$F(u) = \sum_{x=0}^{M-1} f(x)W_M^{ux}, \quad \text{where} \quad W_m := e^{-i2\pi/m} \tag{2.118}$$

We'll define $K \in \mathbb{Z}_+ : 2K = M = 2^n$ (i.e. $K = 2^{n-1}$).

We use this to rewrite the series in eq. (2.118) and split it into odd and even entries in the summation

$$F(u) = \sum_{x=0}^{2K-1} f(x) W_{2K}^{ux} \tag{2.119}$$

$$= \sum_{x=0}^{K-1} f(2x) W_{2K}^{u(2x)} + \sum_{x=0}^{K-1} f(2x+1) W_{2K}^{u(2x+1)} \tag{2.120}$$

We'll get a few identities out of the way (where $m, n, x \in \mathbb{Z}_+$ arbitrary).

$$W_{(2m)}^{(2n)} = e^{\frac{-i2\pi(2m)}{2m}} = e^{\frac{-i2\pi m}{n}} = W_m^n \tag{2.121}$$

$$W_m^{(u+m)x} = e^{\frac{-i2\pi(u+m)x}{m}} = e^{\frac{-i2\pi unx}{m}} e^{\frac{-i2\pi mx}{m}} = e^{\frac{-i2\pi ux}{m}}(1) = W_m^{ux} \tag{2.122}$$

$$W_{2m}^{(u+m)} = e^{\frac{-i2\pi(u+m)}{2m}} = e^{\frac{-i2\pi ux}{2m}} e^{-i\pi} = W_{2m}^u e^{-i\pi} = -W_{2m}^u \tag{2.123}$$

Thus we can rewrite eq. (2.120) as

$$F(u) = \sum_{x=0}^{K-1} f(2x) W_{2K}^{2ux} + \sum_{x=0}^{K-1} f(2x+1) W_{2K}^{2ux} W_{2K}^u \tag{2.124}$$

$$\implies \quad F(u) = \left(\sum_{x=0}^{K-1} f(2x) W_K^{ux} \right) + \left(\sum_{x=0}^{K-1} f(2x+1) W_K^{ux} \right) W_{2K}^u \tag{2.125}$$

The major advance comes via using the identities eq. (2.121) to consider the Fourier transform K frequencies later :

$$F(u+K) = \left(\sum_{x=0}^{K-1} f(2x) W_K^{(u+K)x} \right) + \left(\sum_{x=0}^{K-1} f(2x+1) W_K^{(u+K)x} \right) W_{2K}^{(u+K)} \tag{2.126}$$

$$\implies F(u+K) = \left(\sum_{x=0}^{K-1} f(2x) W_K^{ux} \right) - \left(\sum_{x=0}^{K-1} f(2x+1) W_K^{ux} \right) W_K^u \tag{2.127}$$

Comparing eq. (2.125) and eq. (2.127), we see that the expressions within parentheses are identical. What's more, these parentetical expressions are functionally

identical to discrete fourier transforms themselves. Let's notate them as follows:

$$\mathscr{D}_u\{f_{\text{even}}(t)\} := \sum_{x=0}^{K-1} f(2x) W_K^{ux} \tag{2.128}$$

$$\mathscr{D}_u\{f_{\text{odd}}(t)\} := \sum_{x=0}^{K-1} f(2x+1) W_K^{ux} \tag{2.129}$$

If we're calculating an M point transform (i.e. we're wishing to calculate $F(1),...,F(M)$), once we've calculated the first K discrete frequencies (i.e. $F(1),\cdots,F(K)$) we may simply reuse the two values we've calculated in eq. (2.128) to calculate the next $F(K+1),..,F(K+K) = F(M)$. Since each expression in parentheses involves K complex multiplications and $K-1$ complex additions, we are effectively saving $K(2K-1)$ calculations in computing the entire spectrum $F(1),..,F(M)$. When M is large, the payoff is undeniable.

In fact, through counting calculations and then doing a proof by induction, we can show that the effective number of calculations is given by $M\log_2 M$.

Of course, since eq. (2.128) are DFTs themselves, there's nothing stopping us from reiterating this procedure; if M is substantially large, we can just as easily repeat this process a few times.

Of course, our development was for $1D$. We can extend this to $2D$ by taking note of eq. (2.114).

The one caveat is that the above development was for transforming sequences whose lengths are perfect powers of 2. Since our inputs have no reason to be this, we need to adjust for this. The explanation is that you just do the part that's a power of 2 and then do the rest manually or pick a different power.

Finally we note the inverse DFT can actually be found via a DFT of the complex conjugate of the original signal, and of course we may translate that operation to a FFT.

CHAPTER 3

IMPLEMENTATIONS

Calculating the Hessian

Pseudocode for `np.gradient` which is used in calculating Hessian (code below)

```
gaussian_filtered = fftgauss(image, sigma=sigma)

Lx, Ly = np.gradient(gaussian_filtered)

Lxx, Lxy = np.gradient(Lx)

Lxy, Lyy = np.gradient(Ly)
```

CHAPTER 4

RESEARCH PROTOCOL

Samples / Image Domain

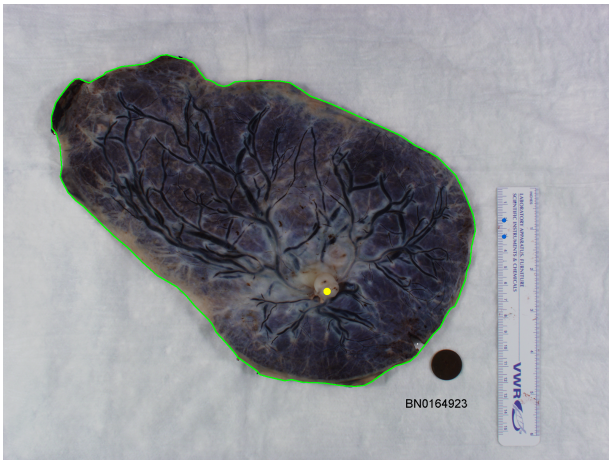
We ultimately perform a PCSVN extraction on a subset of 201 color placental images from a private database provided by the National Children's Study, which had been prepared for a different study. A detailed description of the data set is given in [5], and a description of the cleaning and fixing procedure is given in [2]. The samples are provided as XCF files (the native project file for GIMP) and contain four major layers.

A representative sample

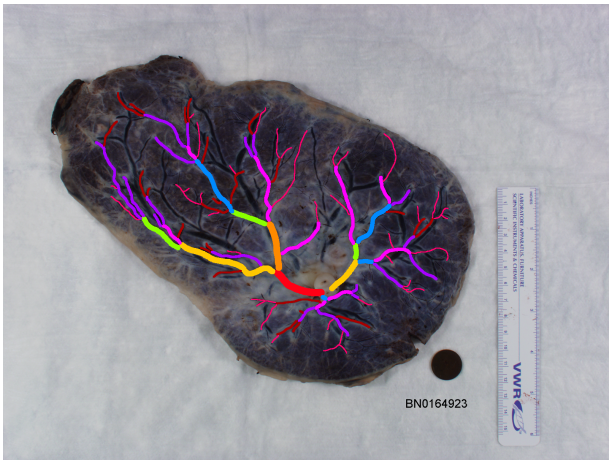
The layers together give a hand tracing of the vascular network and perimeter. A sample of overlaid layers in a representative sample (with ID number "BN0164923") is given in fig. 4.

Each layer is roughly 1954x1200 pixels (with some subtle variation). In fig. 4a, a cleaned, fixed placenta is placed on a table with a camera a fixed distance away, and a rule and penny (presumably for redundancy) to aid registration and calibration of the resolution. fig. 4b is a tracing (in green) of the perimeter of the placenta. The point of umbilical cord insertion is notated in yellow. Two cyan marks are placed on consecutive centimeter markings on the ruler (the dots are enlarged and shown as a darker blue here for clarity). fig. 4c and fig. 4d are both hand traces of the PCSVN, with a layer for each the arteries and veins. These layers are simultaneously overlain on the base image infig. 4e. The coloration is meant to indicate the diameter of each vessel. The diameters are binned into 9 discrete widths, odd integers from 3 to 19 pixels. Vessels of smaller diameter are either binned to three or (quite frequently) left untraced. The

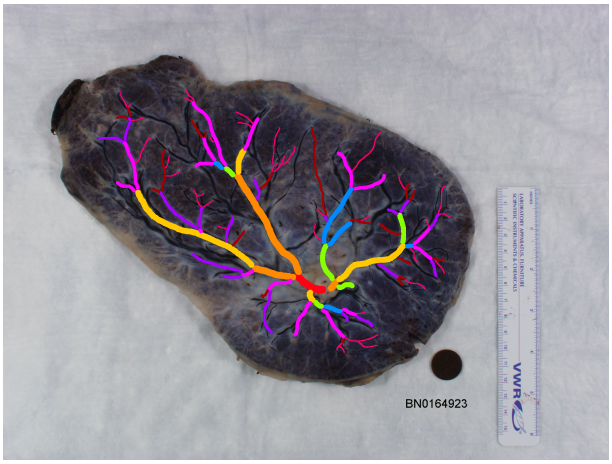
(a) Fixed Placental Sample



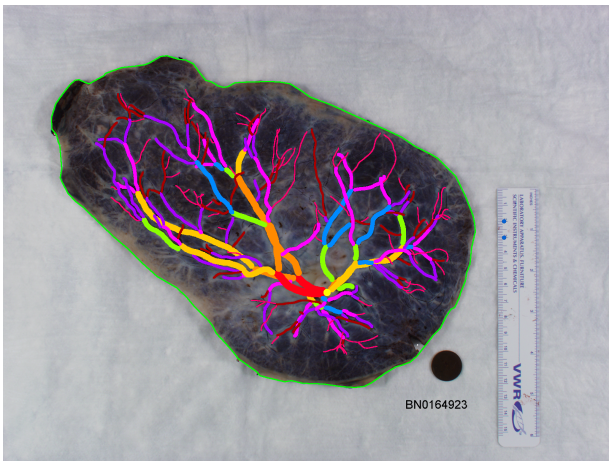
(b) Perimeter tracing



(c) Arterial tracing



(d) Venous tracing



(e) Total Vascular Network

FIGURE 4: A representative placental sample and tracing

vessel width	color (hex value)	color name
3 pixels	#ff006f	magenta
5 pixels	#a80000	dark red
7 pixels	#a800ff	purple
9 pixel s	#ff00ff	light pink
11 pixels	#008aff	blue
13 pixels	#8aff00	green
15 pixels	#ffc800	gold
17 pixels	#ff8a00	orange
19 pixels	#ff0015	bright red

TABLE 1: Vessel width color code

correspondence between pencil color and (binned) vessel width is given in table 1.

All in all, these hand-traced and rather labor intensive–requiring between 4 and 8 hours to trace a single sample. A closer look at many of the samples often reveals that a great deal of subjectivity in providing this "ground truth," as it is not often clear what the underlying truth really is; often it's hard to see where the vein is, vascular networks are obscured by the umbilical stem, the blood in the vessels dries unevenly or ruptures, and the vessel seems to disappear momentarily. These situations and more will be showcased in our results section, where we will discuss methods to simulate the subjectivity of decision.

Knowns and Unknowns

Of course, we wish to simply operate on the placental sample itself, without any understanding of its provided tracing (except for judging the strength of our algorithm); our goal is to develop an algorithm that can produce a "ground truth" tracing similar to fig. 4e or fig. 5d without any user intervention.

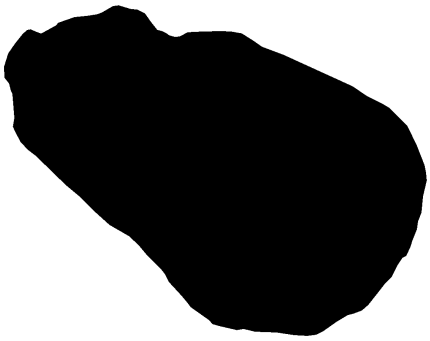
For our purposes however, we will use a limited amount of information from the tracings, namely the provided placental perimeter (shown in green in fig. 4). In developing a fully automated algorithm, it would be relatively straightforward to obtain this boundary ourselves using various techniques, such as an Active Contour Model [26] or, or even a simple edge finding algorithm followed by watershedding and largest object selection as in [13]. We leave that for future work. We do use the traced placental perimeter at our own peril, however, since often there are tears in the side of the plate or large amounts of non-vascular content with large changes in height that are not adequately accounted for in the perimeter tracing.

Finally, we will consider the location of the umbilical insertion point as a "known", as the vessels around it are frequently impossible to see and we wish to exclude them from consideration. It is not unreasonable, however, to consider this to be a known–in future preparations of samples, we could simply require that this point be centered in image in a predictable location. Furthermore, we use its location as a convenience in data analysis–knowledge of this point does not inform our algorithm at all.

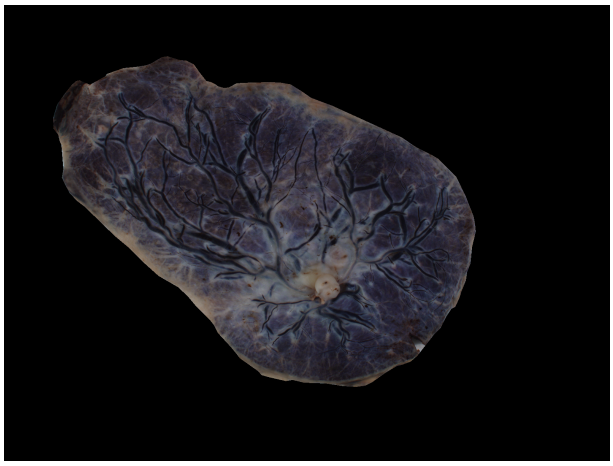
Data Cleaning and Preprocessing

Building a sample suitable for use in our algorithm from fig. 4 is relatively simple. We zero outside the boundary of the plate (so as to not waste computational time calculating the differential geometry of a ruler, say), and also generate a binary mask to identify the plate. Finally, our vessel layers are combined and given as a binary trace. Our preprocessed samples used by the algorithm are given in section 4.2.

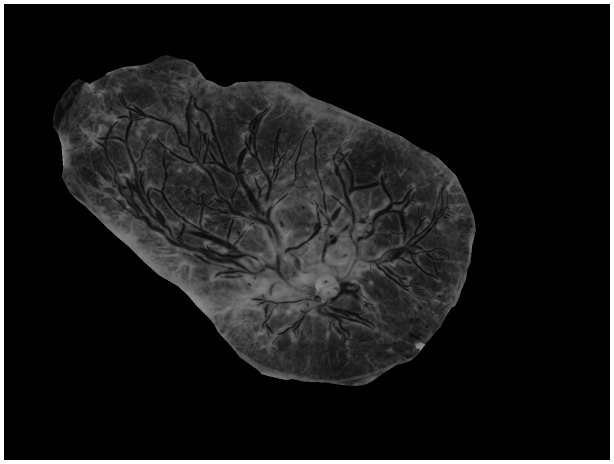
These procedures are performed automatically on the 201 images in our data set using a custom GIMP plug-in, which performs various "bucket fill" operations, layer mergings, and thresholdings. For completeness sake, this plug-in (and an associated Scheme script which turns it into a batch operation) can be found in the Appendix.



(a) Background Mask (in white)



(b) Sample with BG removed



(c) Grayscale

(d) Trace / "Ground Truth"

FIGURE 5: Preprocessed files from an NCS sample

As a point of technicality, the grayscale image in fig. 5c is not actually produced directly by the extractor plug-in, but created when the 3 channel RGB image fig. 5b is imported at the start of the algorithm. This grayscale conversion is simply done for ease of analysis on the sample: although the Frangi filter is designed for arbitrary N-dimensional input [10], an image with three color channels does not have 3 spatial dimensions. We therefore simply combine the information in three channels using the well-known and oft-implemented ITU-R 601-2 luma [15], or "luminance" transform:

$$L = \frac{299}{1000} R + \frac{587}{1000} G + \frac{114}{1000} B \tag{4.1}$$

It should be noted that this choice is not automatic–several other attempts have used the green channel unmodified, as in [2] and [13].

Boundary Dilation

All images are grayscale, M,N pixels as a masked array (of type `numpy.ma.MaskedArray`), where pixels outside of the placental region are masked so they will not be considered by the algorithm. However, some standard implementations of algorithms, namely `numpy.gradient and scipy.signal.convolve2d` are not designed to handle masked regions. Although it would be potentially useful to adapt such methods in a way to, say, calculate a gradient or performs a convolution by a "reflection" across an arbitrary closed boundary (as opposed to the edge of the image matrix), we opted instead to "zero out" unwanted background pixels and simply exclude adjacent areas from consideration. This excluding function, `plate_morphology.dilate_plate`, ultimately relies on two functions provided by the Python library `scikit-image` [30]. The first, `skimage.segmentation.find_boundaries()`, takes the mask input (such as fig. 5a) and calculates where differences in a morphological erosion and dilation occur (which should have the same affect as using the perimeter labeled in fig. 4b directly, though we've chosen to not include that in our sample). That boundary itself is then

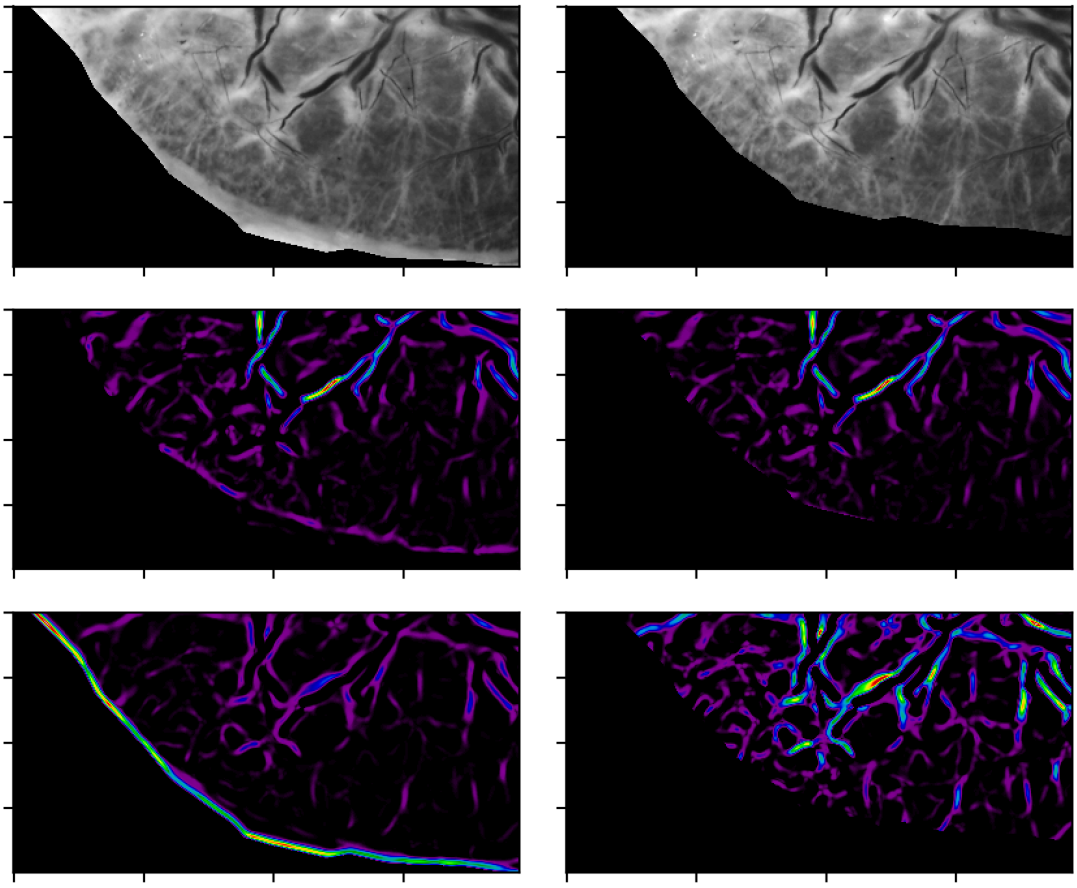


FIGURE 6: Effect of boundary dilation on Frangi responses

dilated by the desired factor. The second is a much quicker implementation of binary dilation that is particularly efficient for our problem: we iterate through an array of indices for the image where the boundary occurs and simply extend the mask R pixels in each direction (like a giant plus sign). Since these pixels are all connected, the effect is very similar to convolving with a disk of radius R, but is much faster.

fig. 6 shows the effect of this so-called "boundary dilation." In the image above, $\sigma = 3$ and border radius is 25 to exaggerate the effect. The first row shows the unaltered boundary of the sample (left) and the sample after boundary dilation (with radius dilation of 25 pixels). The second row shows the Frangi vesselness measure at single

scale ($\sigma = 3$) where `DARK_BG=False` to target dark curvilinear structures performed on the altered sample (left) and the boundary dilated sample (right). Removing an unnecessary part of the placental plate prevents a small response to a non-vascular yet mildly curvilinear background feature from appearing. The third row of fig. 6 shows the Frangi vesselness measure at the same scale ($\sigma = 3$) when we are probing for bright curvilinear structures (i.e. `DARK_BG=True`. Here, wherever the very edge of the placental plate is *any* brighter than adjacent interior, a very large Frangi response will occur, as seen on the left. Dilating the boundary completely avoids this issue, as seen by the figure on the right. Thus we prevent a visual artifact that is present in much prior work on this problem (see [13], [2]). It should be noted that, while the figure on the right shows a much larger interior response, this is simply because the intensity of the output in each of these images is being independently scaled between the minimum and maximum intensity in the image. However, we argue that this is an appropriate and desired depiction of the situation, as we will frequently consider only the relative maxima of Frangi response per scale in our analysis.

We end our discussion by noting that we perform this boundary dilation within the Frangi algorithm itself when we set the structureness parameter γ as half of the maximum Hessian norm found at that scale–this ensures that the maximum occurs sufficiently away from the boundary of the plate.

The code for generating fig. 6 is found in the within the "`if __name__ == __main__`" block of the file `plate_morphology.py`, (so the figure will be generated when running `plate_morphology.py` as a top-level script from the command line). See appendix.

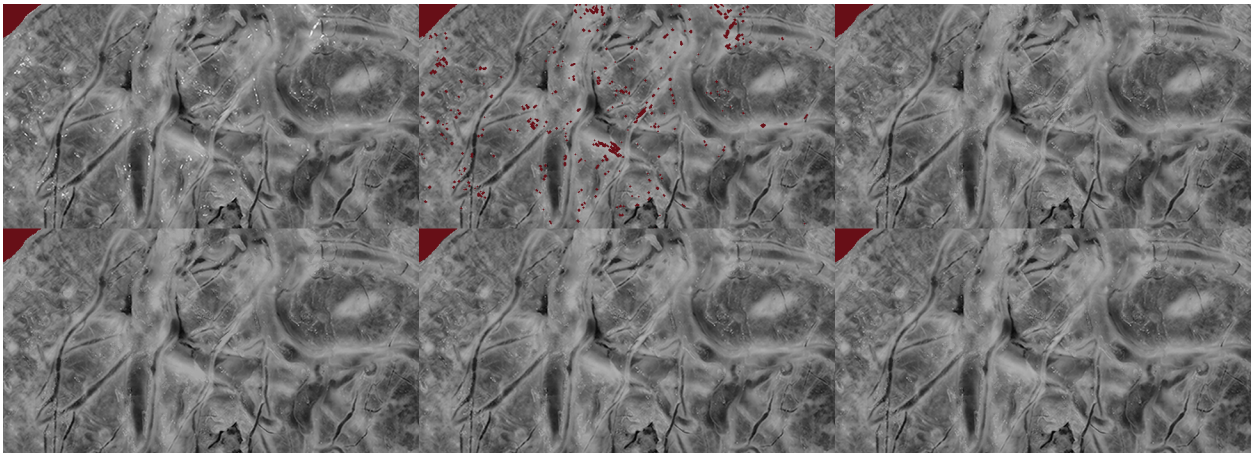


FIGURE 7: Deglaring a sample using a hybrid inpainting method

Deglaring

Despite best efforts when harvesting samples, some placental images have substantial glare, which leads to inaccuracies in identifying curvilinear content. Our protocol for deglaring is analogous to that performed in [2] and [13]. Unfortunately, the method relied upon by those previous papers (`MATLAB`'s `imfill`, which relies on inpainting by solving the Dirichlet problem for masked regions) was not immediately available in a Python environment. Instead, we used an already implemented inpainting algorithm, `scikit-image`'s `inpaint_biharmonic()`, which should be expected to achieve similar results, at the expense of processing time.

The function `inpaint_biharmonic` is based on [7], and relies on solving a biharmonic equation i.e. $\nabla\nabla f = 0$ for the surface f subject to boundary conditions (as compared to `imfill`'s solving the Laplace equation $\nabla f = 0$ in regions marked as glare.

The method for deciding what is considered glare is similar to [2], in which we consider any intensities close the maximum intensity in the image (Almoussa et al. used 80% of max intensity, and we use $175/255 \approx 68\%$). This threshold is dependent on the image domain.

Inpainting in the above way is rather resource intensive, so we implemented two

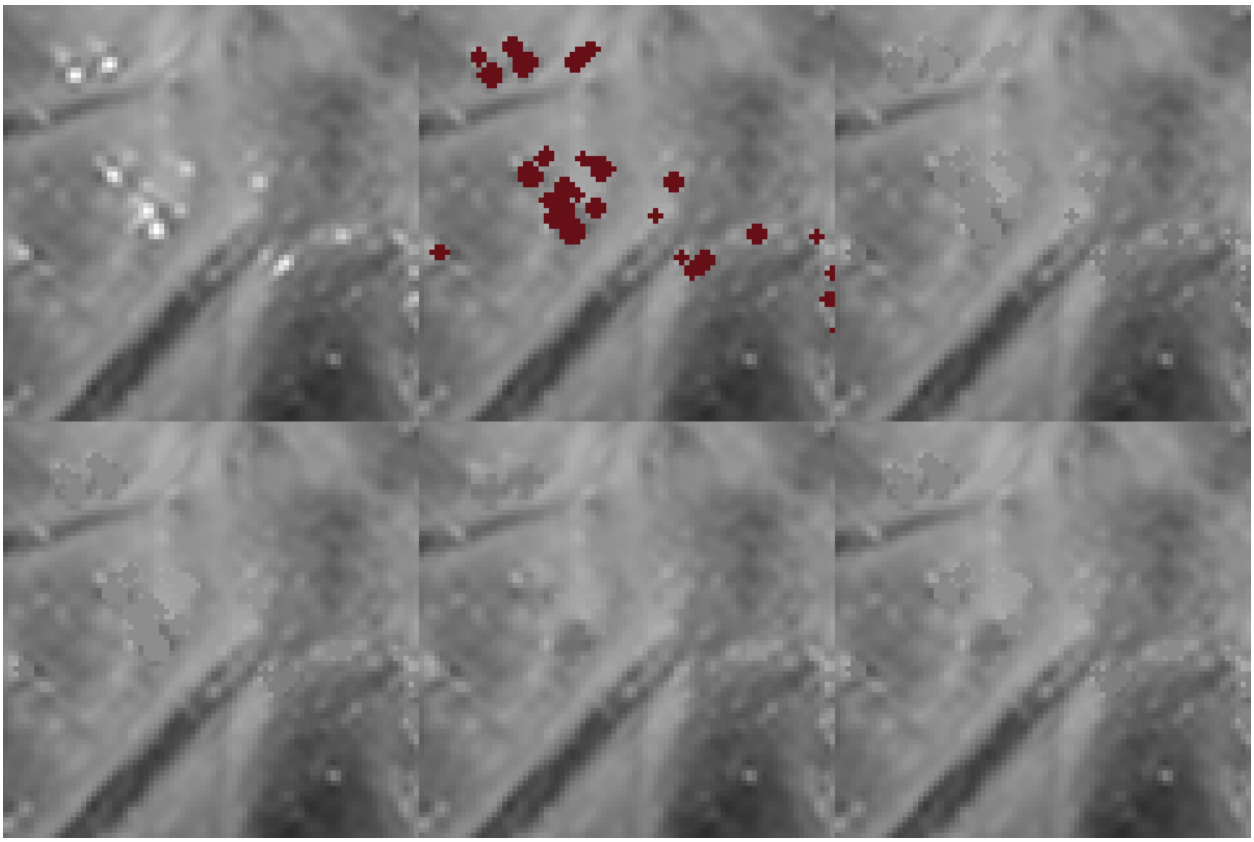


FIGURE 8: Comparison of glare inpainting methods (detail)

faster and less precise methods of inpainting, which can be found in `preprocessing.py`. The first, called `inpaint_glare()` (change this name) replaces any masked pixel with the average of all non-masked values within a certain distance (default 15 pixels). The second, called `inpaint_with_boundary_median` calculates the median value of the (non-masked) boundary and fills any masked region with that value. We argue that these less-exact methods are adequate for smaller regions, while larger regions of glare deserve a more thoughtful application of inpainting. Our final method of inpainting, `inpaint_hybrid` implements this idea–smaller glare regions are inpainted with a boundary median, while larger areas are inpainted with the more expensive but more accurate biharmonic inpainting.

A comparison of these methods is shown in fig. 7, and a zoomed in portion is shown for in fig. 8. In the top left, the glary image is shown. In the top middle, regions above the threshold intensity are masked (shown in dark red, along with the background). In the top right, the strategy is "mean window" with a window size of 15 pixels. The bottom left uses "boundary median" strategy. The middle is the more expensive "biharmonic inpainting" strategy, and the bottom right uses a "hybrid" strategy.

We stress again that only a small subset our image domain exhibits disruptive amounts of glare. Future improvements in this direction should probably seek to implement more robust method such as [18] that is not dependent on an arbitrary threshold for deciding what regions exhibit glare.

Variations in the Data Set and Imperfections of the Ground Truth

Where should this go?

Multiscale Setup

Our multiscale Frangi filter requires a list of scales at which to probe. Each scale is chosen to accentuate features of a particular size, i.e. vessels of a particular radius. This list of scales is denoted as $\Sigma := \{\sigma_1, \sigma_2, \ldots, \sigma_N\}$.

The smallest one should be an effective size where details are expected to be found, and the largest should be an effective size as well. In fact, following [16] it is reasonable and natural to select these logarithmically; that is, for some selected inputs $m < M$ we have

$$\sigma_1 = 2^m \ , \ \sigma_j = 2^{\left(m + \frac{M-m}{N-1} j\right)} \ , \ \sigma_N = 2^M \tag{4.2}$$

That is, the exponents are spaced linearly from m to M. This is achieved by the command `np.logspace(m,M,num=N)`. The idea is that the filter will respond better at its particular scale, but there are diminishing returns as σ increases. While the filter's response may vary substantially between, say $\sigma = 2$ and $\sigma = 3$, there will be not be a substantial difference in response between, say, $\sigma = 46$ and $\sigma = 47$. There was an earlier benefit as well, that is still worth mentioning for historical reasons. Previously, computing the vesselness measure was very expensive, and thus it was simply not feasible to collect so many large scale readings. This is moot with the development of FFT-based Frangi filter.

If there is no particular care taken in selecting a minimum and maximum range at which to probe, then we should assure that there is no noise being introduced at either ends, especially if the Frangi filter at which "throw out" bad ones somehow. We will approach this issue in our discussion of "variable thresholding."

Convolve this via fft transform to get L_{σ_i}

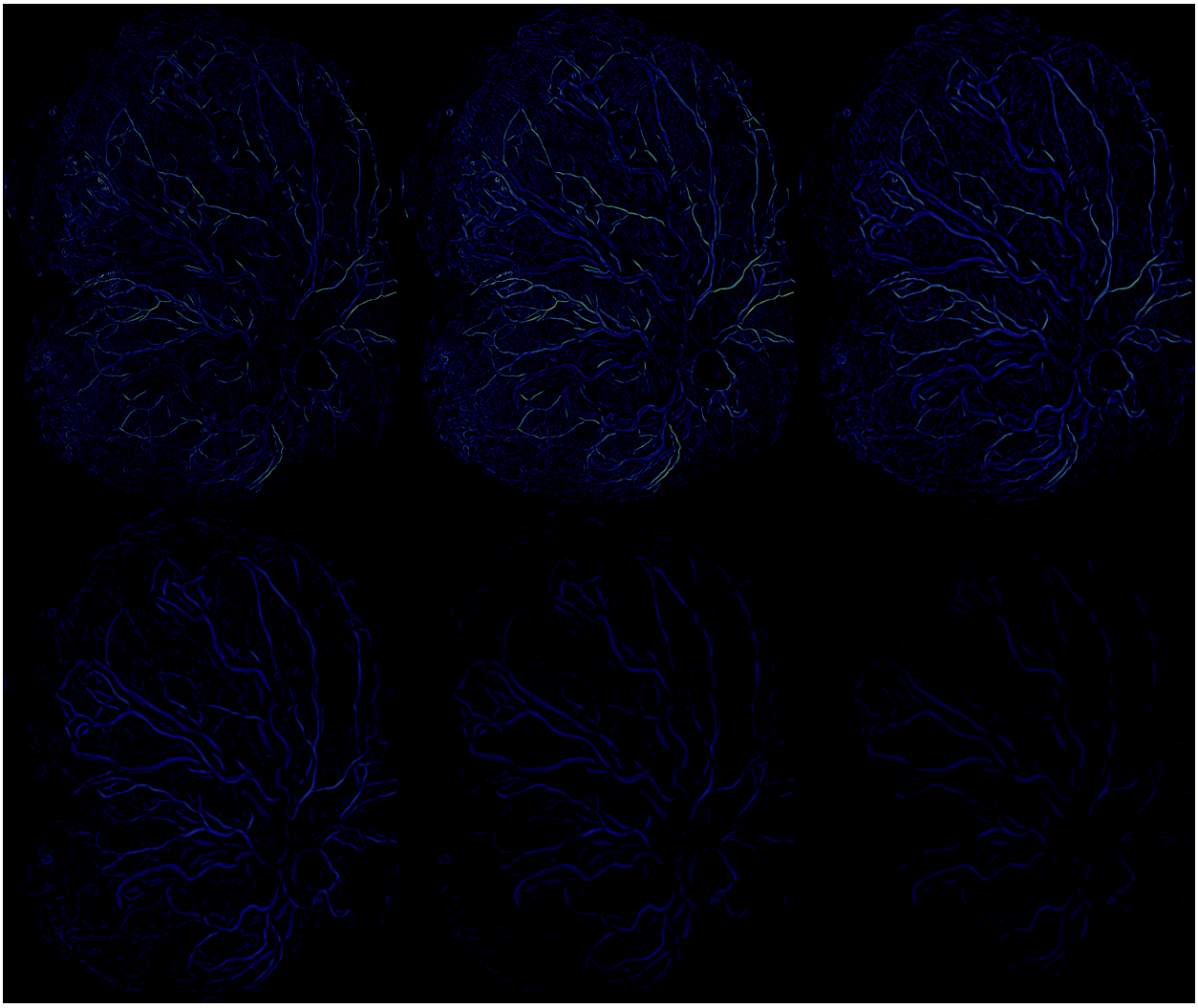


FIGURE 9: Frangi vesselness score at several scales

Applying Vesselness Measure

Calculate the Hessian matrix of and then the eigenvalues using the function `hfft.fft_hessian`.

Scale-space post-processing

Multiscale Merging

Cleanup/Postprocessing

Measurements

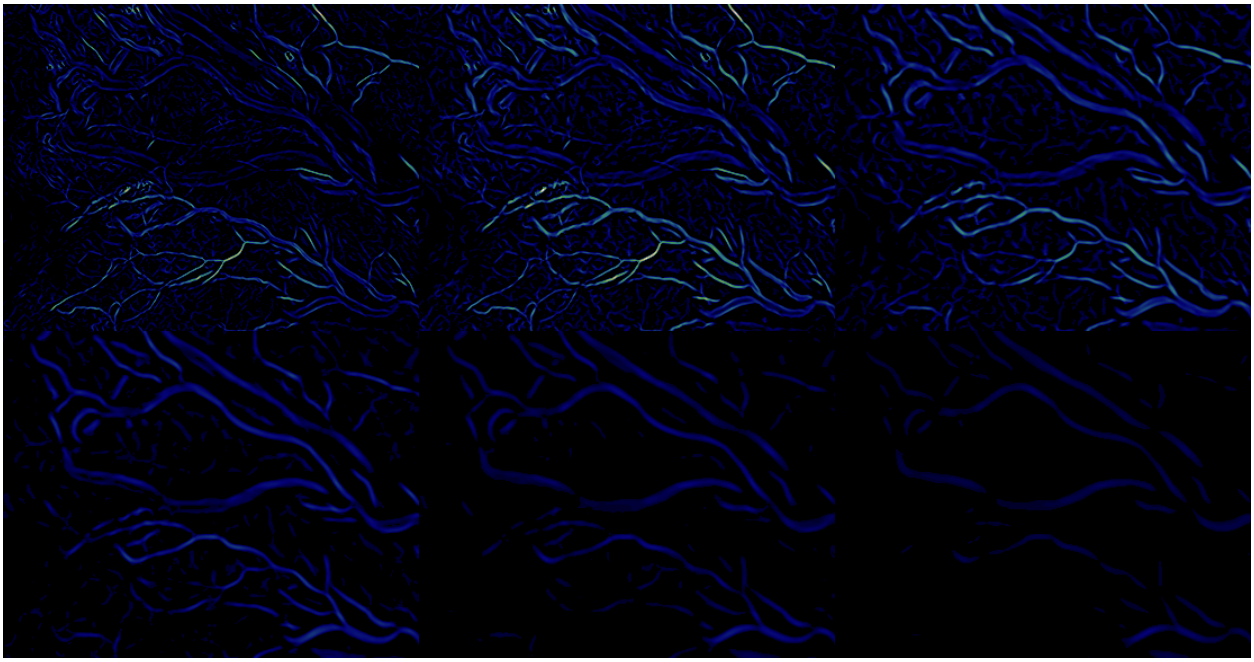


FIGURE 10: Frangi vesselness score at several scales (inset)

Erode plate / dilate boundary

Our function considers the placenta as a nonzero surface but the surface outside is zero (or, in many situations, masked). We're currently not implementing any way to "reflect" along the border, so instead the second degree behavior of the surface there will be incorrect in an area proportional to the scale size.

Describe how that function works. Earlier efforts are wrong, whatever.

The are which is affected should be larger than just the standard dropoff the gaussian however, since we're interested in second derivative information.

CHAPTER 5

RESULTS AND ANALYSIS

use MCC [25]

Sample visual output

The confusion matrix

A Source of "False Negatives" in the NCS data set

Sometimes the output doesn't agree with the trace, i.e. "the ground truth" is not 100% correct. sometimes either there's a false negative (reported) but something just wasn't traced in the original 1602443.

1. Collar is stupid and should really be considered like a error in marking the perimeter. Throw these away or edit. Maybe make a section called discarded samples that's stupid but yeah.

2. Vessels suck sometimes. In the portion above, 1602443, there's a random blood clot which gets identified at large σ. But also the small forked shaped thing which is obviouslly a vessel doesn't get defined.

3. Too much blood (not enough?? no idea) is left in the vessels. leading to the weird white border around some vessels. you could identify these along with black center and combine them somehow. no idea.Also, holy shit, some of the white vessel "sleeves" ARE identified in the tracing, and some aren't. Find an example of this and whine about it.

4. Umbillical cord insertion point is stupid and obscures a lot. The tracer guesses but there's no real guiding principle AFAIK..

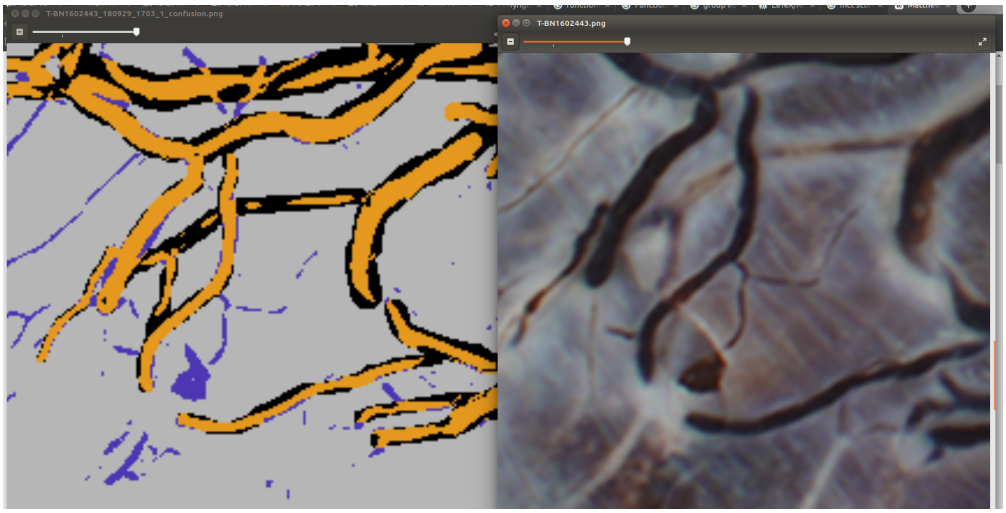


FIGURE 11: ”True” false positives and “False” false positives

5. Small vessels aren’t accounted for at all. Not sure how to coincide measurement in terms of scale space anymore, but should figure out how to cut off those values before running MCC metric.

Results

Answer Research Questions

CHAPTER 6

CONCLUSION

Brief recap.

Review of Work

Estimation of success. Areas of success and struggle.

Future research directions

- Solve the Network Connection Problem (PICTURE OF GAPS) Try something like [19].
- Refine variable thresholding and automate.
- Combine with a ridge search.
- Use this as pre-processing for a Neural Network or something (cite kara's work, katalinas work)
- Apply to more image domains (STARE, WORSE PLACENTAS, ETC.)
- Automate Measurements (more quantative results too)
- Optimize; Better Use of Scales
- Use of Color Data

APPENDICES

APPENDIX A

CODE LISTINGS

The following python scripts and modules were developed with the following packages:

- `python 3.6`
- `numpy`, version 1.12.0
- `scipy`, version 0.19.0
- `scikit-image`, version 0.13.0
- `matplotlib`, version 2.02

Earlier versions of these packages may be compatible but are not guaranteed to be so.

The scripts listed in this appendix are also hosted at `github.com/wukm/pycake`.

listings/pcsvn.py

```
#!/usr/bin/env python3

from placenta import get_named_placenta
from hfft import fft_hessian
from diffgeo import principal_curvatures, principal_directions
from frangi import get_frangi_targets
from skimage.util import img_as_float
import numpy as np
from preprocessing import (inpaint_glare, inpaint_with_boundary_median,
                           inpaint_hybrid)

from plate_morphology import dilate_boundary

import matplotlib.pyplot as plt
import matplotlib as mpl
import numpy.ma as ma

import os.path
import json
import datetime


def make_multiscale(img, scales, betas, gammas,
                    find_principal_directions=False, dilate_per_scale=True,
                    signed_frangi=False, dark_bg=True, kernel=None,
                    VERBOSE=True):
    """Returns an ordered list of dictionaries for each scale of Frangi info.

    Each element in the output contains the following info:
        {'sigma': sigma,
         'beta': beta,
         'gamma': gamma,
```

```
        'H': hesh,
        'F': targets,
        'k1': k1,
        'k2': k2,
        't1': t1, # if find_principal_directions
        't2': t2 # if find_principal_directions
        }

    """

    # store results of each scale (create as empty list)
    multiscale = list()

    img = ma.masked_array(img_as_float(img), mask=img.mask)

    for i, sigma, beta, gamma in zip(range(len(scales)), scales,
                                     betas, gammas):
        if dilate_per_scale:
            if sigma < 2.5:
                radius = 10
            elif sigma > 20:
                radius = int(sigma*2)
            else:
                radius = int(sigma*4)  # a little aggressive
        else:
            radius = None

        if VERBOSE:
            print('  ={}'.format(sigma))

        # get hessian components at each pixel as a triplet (Lxx, Lxy, Lyy)
        hesh = fft_hessian(img, sigma, kernel=kernel)

        if VERBOSE:
            print('finding principal curvatures')

        # calculate principal curvatures with |k1| <= |k2|
        k1, k2 = principal_curvatures(img, sigma=sigma, H=hesh)

        # area of influence to zero out
        if dilate_per_scale:
            collar = dilate_boundary(None, radius=radius, mask=img.mask)

            k1[collar] = 0
            k2[collar] = 0

        # set anisotropy parameter if not specified
        if gamma is None:
            # Frangi suggested 'half the max Hessian norm' as an empirical
            # half the max spectral radius is easier to calculate so do that
            # shouldn't be affected by mask data but should make sure the
            # mask is *well* far away from perimeter
            # we actually calculate half of max hessian norm
            # using frob norm = sqrt(trace(AA^T))
            hxx, hxy, hyy = hesh
            hessian_norm = np.sqrt((hxx**2 + 2*hxy**2 + hyy**2))

            # make sure the max doesn't occur on a boundary
            dilation_radius = int(max(np.ceil(sigma), 10))
            collar = dilate_boundary(None, radius=dilation_radius,
                                     mask=img.mask)
            hessian_norm[collar] = 0
            max_hessian_norm = hessian_norm.max()
            gamma = .5*max_hessian_norm
```

```
97
98              if VERBOSE:
99                  # compare to other method of calculating gamma
100                 gamma_alt = .5 * np.abs(k2).max()
101                 print(f"half of k2 max is {gamma_alt}")
102
103         if VERBOSE:
104             print(f"gamma (half of max hessian (frob) norm is {gamma}")
105             print(f'finding Frangi targets with   ={beta} and   ={gamma:.2}')
106
107         # calculate frangi targets at this scale
108         targets = get_frangi_targets(k1, k2, beta=beta, gamma=gamma,
109                                      dark_bg=dark_bg, signed=signed_frangi)
110
111
112         # store results as a dictionary
113         this_scale = {'sigma': sigma,
114                       'beta': beta,
115                       'gamma': gamma,
116                       'H': hesh,
117                       'F': targets,
118                       'k1': k1,
119                       'k2': k2,
120                       'border_radius': radius
121                       }
122
123         if find_principal_directions:
124             # principal directions should only be computed for critical regions
125             # ignore anything less than a std deviation over the mean
126             # this mask is where PD's will *NOT* be calculated
127             cutoff = targets.mean() + targets.std()
128             pd_mask = np.bitwise_or(targets < cutoff, img.mask).filled(1)
129             percent_calculated = (pd_mask.size - pd_mask.sum()) / pd_mask.size
130
131             if VERBOSE:
132                 print(f"finding PD's for {percent_calculated:.2%} of image"
133                       f"anything above vesselness score {cutoff:.6f}"
134                       )
135             t1, t2 = principal_directions(img, sigma=sigma, H=hesh,
136                                           mask=pd_mask)
137
138             # add them to this scale's output
139             this_scale['t1'] = t1
140             this_scale['t2'] = t2
141         else:
142             if VERBOSE:
143                 print('skipping principal direction calculation')
144
145         # store results as a list of dictionaries
146         multiscale.append(this_scale)
147
148     return multiscale
149
150
151 def extract_pcsvn(filename, scales, alphas=None, betas=None, gammas=None,
152                   DARK_BG=True, dilate_per_scale=True, verbose=True,
153                   generate_json=True, output_dir=None, kernel=None,
154                   signed_frangi=False, remove_glare=False):
155     """Run PCSVN extraction on the sample given in the file.
156
157     Despite the name, this simply returns the Frangi filter responses at
158     each provided scale without explicitly making any decisions about what
159     is or is not part of the PCSVN.
160
```

```
    TODO:Finish docstring!
    """

    raw_img = get_named_placenta(filename, maskfile=None)

    # Multiscale & Frangi Parameters######################

    # set default alphas and betas if undeclared
    if alphas is None:
        alphas = [.15 for s in scales]  # threshold constant
    if betas is None:
        betas = [0.5 for s in scales]  # anisotropy constant

    # declare None here to calculate half of hessian's norm
    if gammas is None:
        gammas = [None for s in scales]  # structureness parameter

    # Preprocessing###############
    if remove_glare:
        if verbose:
            print('removing glare from sample')
        # img = inpaint_glare(raw_img)
        # img = inpaint_with_boundary_median(raw_img)
        img = inpaint_hybrid(raw_img)
    else:
        img = raw_img.copy()  # in case we alter the mask or something

    # Multiscale Frangi Filter###############################

    # output is a dictionary of relevant info at each scale
    multiscale = make_multiscale(img, scales, betas, gammas,
                                 find_principal_directions=False,
                                 dilate_per_scale=dilate_per_scale,
                                 kernel=kernel,
                                 signed_frangi=signed_frangi,
                                dark_bg=DARK_BG,
                                 VERBOSE=verbose)

    # extract these for logging
    gammas = [scale['gamma'] for scale in multiscale]
    border_radii = [scale['border_radius'] for scale in multiscale]

    ###Process Multiscale Targets############################

    # ignore targets too close to edge of plate
    # wait are we doing this twice?
    if dilate_per_scale:
        if verbose:
            print('trimming collars of plates (per scale)')

        for i in range(len(multiscale)):
            f = multiscale[i]['F']
            # twice the buffer (be conservative!)
            radius = int(multiscale[i]['sigma']*2)
            if verbose:
                print('dilating plate for radius={}'.format(radius))
            f = dilate_boundary(f, radius=radius, mask=img.mask)
            # get rid of mask
            multiscale[i]['F'] = f.filled(0)
    else:
        for i in range(len(multiscale)):
            # get rid of mask
            multiscale[i]['F'] = multiscale[i]['F'].filled(0)

```

```
    ###Make Composite###########################################

    # get a M x N x n_scales array of Frangi targets at each level
    F_all = np.dstack([scale['F'] for scale in multiscale])

    if generate_json:

        time_of_run = datetime.datetime.now()
        timestring = time_of_run.strftime("%y%m%d_%H%M")

        logdata = {'time': timestring,
                'filename': filename,
                'alphas': list(alphas),
                'betas': list(betas),
                'gammas': gammas,
                'sigmas': list(scales),
                }
        if dilate_per_scale:
            logdata['border_radii'] = border_radii

        if output_dir is None:
            output_dir = 'output'

        base = os.path.basename(filename)
        *base, suffix = base.split('.')
        dumpfile = os.path.join(output_dir,
                                ''.join(base) + '_' + str(timestring)
                                + '.json')

        with open(dumpfile, 'w') as f:
            json.dump(logdata, f, indent=True)

    # this function used to returns scales and alphas too but now doesn't,
    return F_all, img


def get_outname_lambda(filename, output_dir=None, timestring=None):
    """
    return a lambda function which can build output filenames
    """

    if output_dir is None:
        output_dir = 'output'

    base = os.path.basename(filename)
    *base, suffix = base.split('.')

    if timestring is None:
        time_of_run = datetime.datetime.now()
        timestring = time_of_run.strftime("%y%m%d_%H%M")

    outputstub = ''.join(base) +'_' + timestring +  '_{}.'+ suffix
    return lambda s: os.path.join(output_dir, outputstub.format(s))


def _build_scale_colormap(N_scales, base_colormap, basecolor=(0,0,0,1)):
    """
    returns a mpl.colors.ListedColormap with N samples,
    based on the colormap named "default_colormap" (a string)

    the N colors are given by the default colormap, and
    basecolor (default black) is added to map to 0.
    (you could change this, for example, to (1,1,1,1) for white)

```

```
    reversed colormaps often work better if the basecolor is black
    you should make sure there's good contrast between the basecolor
    and the first color in the colormap
    """

    map_range = np.linspace(0, 1, num=N_scales)

    colormap = plt.get_cmap(base_colormap)

    colorlist = colormap(map_range)

    # add basecolor as the first entry
    colorlist = np.vstack((basecolor, colorlist))

    return mpl.colors.ListedColormap(colorlist)

def scale_label_figure(wheres, scales, savefilename=None,
                       crop=None, show_only=False, image_only=False,
                       save_colorbar_separate=False, savecolorbarfile=None,
                       output_dir=None):
    """
    crop is a slice object.
    if show_only, then just plt.show (interactive).
    if image_only, then this will *not* be printed with the colorbar

    if save_colormap_separate, then the colormap will be saved as a separate
    file
    """
    if crop is not None:
        wheres = wheres[crop]

    fig, ax = plt.subplots() # not sure about figsize
    N = len(scales) # number of scales / labels

    tabemap = _build_scale_colormap(N, 'viridis_r')

    if image_only:
        plt.imsave(savefilename, wheres, cmap=tabemap, vmin=0, vmax=N)
        plt.close()
    else:
        imgplot = ax.imshow(wheres, cmap=tabemap, vmin=0, vmax=N)
        # discrete colorbar
        cbar = plt.colorbar(imgplot)

        # this is apparently hackish, beats me
        tick_locs = (np.arange(N+1) + 0.5)*(N-1)/N

        cbar.set_ticks(tick_locs)
        # label each tick with the sigma value
        scalelabels = [r"$\sigma = {:.2f}$".format(s) for s in scales]
        scalelabels.insert(0, "(no match)")
        # label with their sigma value
        cbar.set_ticklabels(scalelabels)
        #ax.set_title(r"Scale ($\sigma$) of maximum vesselness ")
        plt.tight_layout()

        #plt.savefig(outname('labeled'), dpi=300)
        if show_only or (savefilename is None):
            plt.show()
        else:
            plt.savefig(savefilename, dpi=300)

        plt.close()

```

```
    if save_colorbar_separate:
        if savecolorbarfile is None:
            savecolorbarfile = os.path.join(output_dir, "scale_colorbar.png")
        fig = plt.figure(figsize=(1, 8))
        ax1 = fig.add_axes([0.05, 0.05, 0.15, 0.9])
        tick_locs = (np.arange(N+1) + 0.5)*(N-1)/N
        scalelabels = [r"$\sigma = {:.2f}$".format(s) for s in scales]
        scalelabels.insert(0, "n/a")
        cbar = mpl.colorbar.ColorbarBase(ax1, cmap=tabemap,
                                         norm=mpl.colors.Normalize(vmin=0,
                                                                   vmax=N),
                                         orientation='vertical',
                                         ticks=tick_locs)
        cbar.set_ticklabels(scalelabels)
        plt.savefig(savecolorbarfile, dpi=300)
```

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover, New York, ninth dover printing, tenth gpo printing edition, 1964.

[2] Nizar Almoussa, Brittany Dutra, Bryce Lampe, Pascal Getreuer, Todd Wittman, Carolyn Salafia, and Luminita Vese. Automated vasculature extraction from placenta images. In *Medical Imaging 2011: Image Processing*, volume 7962, page 79621L. International Society for Optics and Photonics, 2011.

UCLA REU

[3] J. Babaud, M. Baudin, R. O. Duda, and A. P. Witkin. Uniqueness of the gaussian kernel for scale-space filtering. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 8:26–33, 01 1986.

[4] R Burden and J Faires. *Numerical Analysis*. Brooks/Cole, 9 edition, 2011.

[5] Jen-Mei Chang, Hui Zeng, Ruxu Han, Ya-Mei Chang, Ruchit Shah, Carolyn M Salafia, Craig Newschaffer, Richard K Miller, Philip Katzman, Jack Moye, et al. Autism risk classification using placental chorionic surface vascular network features. *BMC medical informatics and decision making*, 17(1):162, 2017.

[6] Ya-Mei Chang, Ruxu Han, Hui Zeng, Ruchit Shah, Craig Newschaffer, Richard Miller, Philip Katzman, John Moye, Carolyn Salafia, et al. Whole chorionic surface vessel feature analysis with the boruta method, and autism risk. *Placenta*, 45:75, 2016.

[7] SB Damelin and NS Hoang. On surface completion and image inpainting by biharmonic functions: Numerical aspects. *International Journal of Mathematics and Mathematical Sciences*, 2018, 2018.

[8] Karamatou Yacoubou Djima, Carolyn Salafia, Richard K Miller, Ronald Wood, Philip Katzman, Chris Stodgell, and Jen-Mei Chang. Enhancing placental chorionic surface vasculature from barium-perfused images with directional and multiscale methods. *Placenta*, 57:292–293, 2017.

[9] Bengt Fornberg. Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of computation*, 51(184):699–706, 1988.

[10] Alejandro F Frangi, Wiro J Niessen, Koen L Vincken, and Max A Viergever. Multiscale vessel enhancement filtering. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 130–137. Springer, 1998.

[11] Rafael C Gonzalez and Richard E Woods. Digital image processing prentice hall. *Upper Saddle River, NJ*, 2002.

[12] E. Hille and R.S. Phillips. *Functional Analysis and Semi-groups*. American Mathematical Society: Colloquium publications. American Mathematical Society, 1957.

cited within Sporring just for one thing

[13] Nen Huynh. *A filter bank approach to automate vessel extraction with applications*. PhD thesis, California State University, Long Beach, 2013.

[14] Xiangmin Jiao and Hongyuan Zha. Consistent computation of first-and second-order differential quantities for surface meshes. In *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pages 159–170. ACM, 2008.

[15] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed ¡today¿].

[16] Jan J. Koenderink. The structure of images. *Biological Cybernetics*, 50(5):363–370, Aug 1984.

[17] W. Kühnel, B. Hunt, and American Mathematical Society. *Differential Geometry: Curves - Surfaces - Manifolds*. Student mathematical library. American Mathematical Society, 2006.

[18] Holger Lange. Automatic glare removal in reflectance imagery of the uterine cervix. In *Medical Imaging 2005: Image Processing*, volume 5747, pages 2183–2193. International Society for Optics and Photonics, 2005.

[19] Ivan Laptev, Helmut Mayer, Tony Lindeberg, Wolfgang Eckstein, Carsten Steger, and Albert Baumgartner. Automatic extraction of roads from aerial images based on scale space and snakes. *Machine Vision and Applications*, 12(1):23–31, 2000.

[20] Tony Lindeberg. *On the construction of a scale-space for discrete images*. KTH Royal Institute of Technology, 1988.

[21] Tony Lindeberg. Scale-space for discrete signals. *IEEE transactions on pattern analysis and machine intelligence*, 12(3):234–254, 1990.

[22] Tony Lindeberg. Discrete derivative approximations with scale-space properties: A basis for low-level feature extraction. *Journal of Mathematical Imaging and Vision*, 3(4):349–376, 1993.

[23] Tony Lindeberg. Feature detection with automatic scale selection. *International journal of computer vision*, 30(2):79–116, 1998.

[24] C. Lorenz, I. C. Carlsen, T. M. Buzug, C. Fassnacht, and J. Weese. Multi-scale line segmentation with automatic estimation of width, contrast and tangential direction in 2d and 3d medical images. In Jocelyne Troccaz, Eric Grimson, and Ralph Mösges, editors, *CVRMed-MRCAS'97*, pages 233–242, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[25] B.W. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442–451, 1975.

[26] Anca Morar, Florica Moldoveanu, and Eduard Gröller. Image segmentation based on active contours without edges. In *2012 IEEE 8th International Conference on Intelligent Computer Communication and Processing*, pages 213–220. IEEE, 2012.

[27] Sílvia Delgado Olabarriaga, M Breeuwer, and WJ Niessen. Evaluation of hessian-based filters to enhance the axis of coronary arteries in ct images. In *International Congress Series*, volume 1256, pages 1191–1196. Elsevier, 2003.

[28] Yoshinobu Sato, Shin Nakajima, Nobuyuki Shiraga, Hideki Atsumi, Shigeyuki Yoshida, Thomas Koller, Guido Gerig, and Ron Kikinis. Three-dimensional multi-scale line filter for segmentation and visualization of curvilinear structures in medical images. *Medical image analysis*, 2(2):143–168, 1998.

[29] Jon Sporring. *Gaussian Scale-Space Theory*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[30] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014.

[31] J. H. Wilkinson, editor. *The Algebraic Eigenvalue Problem*. Oxford University Press, Inc., New York, NY, USA, 1988.