ABSTRACT

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

By

Lucas Wukmer

May 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 Frangis 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

May 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

May 2018

ACKNOWLEDGEMENTS

Acknowledgments go here.

TABLE OF CONTENTS

APPENDIX Page

LIST OF TABLES

TABLE Page

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The Applied Problem

Reference Nen's paper and latest autism risk paper.

Context in Image Processing

- brief background of math image processing methods
- what's been tried in this applied problem
  - nen [2]
  - catalina's paper
  - kara's paper
  - other domains

Research Goals

Segue from previous paragraph, talk about strengths and weaknesses of other methods and what this research aims to accomplish. Include 'research questions' that could allow a reader to answer the question "will this research work for my problem?". "Elevator pitch" maybe goes here.

Roadmap

Outline of the thesis ("firstly" bullshit)

CHAPTER 2

MATHEMATICAL METHODS

Notes on this draft:

The following types of comments are interspersed throughout this working draft. Obviously, these are to be removed by the time of the final draft.

[TODO: This is a TODO. Used when something is missing, incomplete, or wrong. These are the most important action items, often requiring additional research or development. For example, "add example" or "derive necessity of gaussian" i]

[CLEANUP: Lower priority than a TODO. Cleanup-TODOs can and will require in-depth fixes, but it's usually less pressing than a missing/wrong item, which requires a TODO. For example, "fix transition" or "standardize notation" is a CLEANUP, not a TODO. Roughly, CLEANUPS are for formatting/style concerns, rather than content.]

[This is a comment. Used for general remarks about a subject that haven't been merged into the text. These are often descriptions, or suggestions for expansion or deletion of content.]

These should be used sparing. Absence of any comment does not indicate that a section is complete or without concerns in any way–these are simply meant to represent "threads to pull" in bringing this document closer to completion.

Problem Setup in Image Processing

[This section should be basically describing how to view an image as a surface. Use any notation here that is useful beyond differential geometry, in all the contexts we need to consider the image (in terms of Fourier Theory, scale space theory, Frangi filtering, etc. all together.]

[CLEANUP: Less halfassed intro. Do this last, when the flow of this chapter is fixed.]

Basics, Definitions

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

[CLEANUP: I hate using L since it conflicts with the standard symbol used for the Weingarten map later. Fix these.]

Definition 2.1.1 (Image as a pixel matrix)**.** $\mathtt{I} \in \mathbb{R}^{m \times n}$ *with* $I_{ij} \in 0..255$

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 2.1.2 (Image as an interpolated surface)**.** $h : \mathbb{R}^2 \to \mathbb{R}$ *with* $h \in \mathcal{C}^2\left(\mathbb{R}^2\right)$.

[the following paragraph is stupid probably. acknowledge it elsewhere or destroy it] It is indeed a sampling, and we could note that $h(i, j) = \mathtt{I}_{ij}$. Of course, $\mathtt{I}$ is not a perfect representation of the underlying "content" within the picture. There are multiple treatments of image processing that do address this [CLEANUP: cite], especially when the goal is noise reduction. However, we will be content to simply represent the pixels of definition 2.1.1 as the ultimate "cause" of the surface definition 2.1.2, rather than considering $\mathtt{I}$ to be a sampling of some real-world surface.

[TODO: but this contradicts the motivation behind scale space theory later. rectify]

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 [CLEANUP: cite Kuhnel]

Preliminaries of Differential Geometry / Notational Dump

[CLEANUP: Mention when you're talking about a general surface in $\mathbb{R}^3$ and when it's specifically a graph and make sure it's clear which case you're dealing with and motivate why

you'd want to talk about things that aren't graphs at all (to define shape operator in general). Potentially rework to not talk about non-graphs when irrelevant. Although it's useful to keep some references to non-graphs to align this with "the general picture" and what you'll find in a diff geo textbook, also because I already wasted time doing the hard work.]

Given an open subset $U \subset \mathbb{R}^2$ and a twice differentiable function $h : U \to \mathbb{R}$ (as in definition 2.1.2) we define its *graph* f as

Definition 2.2.1. *The surface f is a graph (of the function h) when*

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

For any point $u \in U$, we associate a $p \in f[U]$, i.e. $p = f(u)$.

[CLEANUP: make these all definitions and fix the flow]

Definition 2.2.2. *We define the tangent plane of f at point p as the map*

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

where [CLEANUP: fix this flow. subdefintions?]

Definition 2.2.3. *$Df|_x$ is the differential map of f at point x given by*

$$Df|_x : U \to \mathbb{R}^3 \quad by \quad v \mapsto J_f(x)\cdot v$$

where [CLEANUP: another definition?] $J_f(x)$ is the Jacobian of f evaluated at point x, i.e. the matrix

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

We shall denote a 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, span $\left\{\frac{\partial f}{\partial u_1}, \frac{\partial f}{\partial u_2}\right\} = T_u f$. *[maybe do this after you finish basic defintions]*

Of course, we should justify this. *[what's the smallest statement you want to give lemma status?]*

Lemma 2.2.1. *For the surface f of a graph at any point p, $\left\{\frac{\partial f}{\partial u_1}, \frac{\partial f}{\partial u_2}\right\}$ is a linearly independent set.*

Proof. This is actually true for any immersion f. We have no need to prove the general case [CLEANUP: define what an immersion is? or just cite], but in our use case, where f is a graph as in definition 2.2.1, the claim is clear: we can directly see that f_{u_1} and f_{u_2} are linearly independent, as $f_{u_1} = (1,0,h_{u_2})$ and $f_{u_2} = (0,1,h_{u_2})$, which are clearly linearly independent. □

The partials derivatives of f are not, in general, orthogonal. So the differential map of f is exactly the expansion of a point $v \in U$ along the basis $\left\{\frac{\partial f}{\partial u_1}, \frac{\partial f}{\partial u_2}\right\}$.

Definition 2.2.4 (Curves on a surface)**.** *Given a closed interval $I \subset \mathbb{R}$, we can define a regular curve on the surface $c : I \to \mathbb{R}^3$ such that* image$(c) \subset$ image(f)*. In other words, we can (for this particular curve) define an intermediary parametrization θ_c for this curve so that $c = f \circ \theta_c$ i.e.*

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

and $c(t) = f(\theta(t))$.

We regularize our concept of a "normal" direction to the surface, which will prove immensely useful later **[TODO: add pictures!]**

Definition 2.2.5 (The Gauss Map)**.** *The Gauss map at a point $p = f(u)$ is the unit normal to the tangent plane*

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

Each partial above understood to be evaluated at the input $u \in U$,that is, we calculate $\frac{\partial f}{\partial u_i}\Big|_u$. It is clear that $\nu \perp \frac{\partial f}{\partial u_i}$ each $i = 1,2$, and also that span $\left\{\frac{\partial \nu}{\partial u_1}, \frac{\partial \nu}{\partial u_2}\right\} \subset T_u f$ as well. *[at this point if you can prove $\left\{\frac{\partial \nu}{\partial u_1}, \frac{\partial \nu}{\partial u_2}\right\}$ are linearly independent then do so, or just save for later, in which case don't use derivates at all]*

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 $\text{span}\left\{\frac{\partial \nu}{\partial u_i}\right\} \parallel \text{span}\left\{\frac{\partial f}{\partial u_i}\right\}$.

Curvature of a surface

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 a curve embedded within the surface: that is, $c[I] = f[\theta_c[I]]$. Considering a point $p \in I$ and its associated point $u = \theta_c(p)$, we wish to compare the curvatures of all curves at the point with a shared velocity. We now present a main result that provides a notion of curvature of a surface.

Theorem 2.2.2 (Theorem of Meusnier)**.** *Given a point $u \in U$ and a tangent direction $X \in T_u f$, any 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.*

[TODO: provide a visualization of this]

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.

Proof. Considering any such curve where $\frac{\partial c}{\partial t}(p) = X$ where $X \in T_u f$ is a normalized vector tangent to the surface at the point u, we wish to decompose the curve's acceleration along the orthogonal vectors X and the Gauss map $\nu := \nu(u_1, u_2) = \frac{\frac{\partial f}{\partial u_1} \times \frac{\partial f}{\partial u_2}}{\|\frac{\partial f}{\partial u_1} \times \frac{\partial f}{\partial u_2}\|}$. (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 $u = \theta_c(p)$)

$$c'' = \langle c'', X \rangle X + \langle c'', \nu \rangle \nu \tag{2.1}$$

.

The first term is always zero for a regular curve:

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

since either $c'' \perp c'$ (which is generally true for a regular curve in $\mathbb{R}^3$ with a nontrivial curvature) or $c'' = 0$ itself. *[see Kuhnel pg. 13, def 2.4. What results of differential geometry are too basic? what's off limits? Should I develop this too?]*

We can rewrite the second coefficient of eq. (2.1) using the chain rule:

$$\begin{aligned}
\langle c'', \nu \rangle &= \frac{\partial}{\partial t} \left[\langle c', \nu \rangle \right] - \langle c', \frac{\partial \nu}{\partial t} \rangle && (2.2) \\
&= \frac{\partial}{\partial t} \left[\langle X, \nu \rangle \right] - \langle c', \frac{\partial \nu}{\partial t} \rangle && (2.3) \\
&= 0 - \langle X, \frac{\partial \nu}{\partial t} \rangle && (2.4)
\end{aligned}$$

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

$$\begin{aligned}
\|c''\| = \|\langle c'', X \rangle X + \langle c'', \nu \rangle \nu\| &= \|0 + \langle c'', \nu \rangle \nu\| && (2.5) \\
&= -\langle X, \frac{\partial \nu}{\partial t} \rangle \|\nu\| && (2.6) \\
&= -\langle X, \frac{\partial \nu}{\partial t} \rangle && (2.7) \\
&= \langle X, -\frac{\partial \nu}{\partial t} \rangle && (2.8)
\end{aligned}$$

We may compute $-\frac{\partial \nu}{\partial t}$ via chain rule:

$$\begin{aligned}
-\frac{d\nu}{dt} &= -\frac{d}{dt} \left[\nu(u_1, u_2) \right] && (2.9) \\
&= -\frac{d}{dt} \left[\nu(\theta_1(t), \theta_2(t)) \right] && (2.10) \\
&= \theta_1'(t) \left(-\frac{\partial \nu}{\partial u_1} \right) + \theta_2'(t) \left(-\frac{\partial \nu}{\partial u_2} \right) && (2.11)
\end{aligned}$$

Identifying $\left\{ \frac{\partial \nu}{\partial u_i} \right\}_{i=1,2}$ as a subset of $T_u f$, we can identify a linear transformation $\mathsf{L} : T_u f \to T_u f$ which maps the basis $\left\{ \frac{\partial f}{\partial u_i} \right\}_{i=1,2}$ to this subset, i.e. $\mathsf{L}(\frac{\partial f}{\partial u_i}) = -\frac{\partial \nu}{\partial u_i}$. This allows us

to rewrite the time derivative of the Gauss map eq. (2.9) as

$$-\frac{d\nu}{dt} = \theta_1'(t)\left(-\frac{\partial \nu}{\partial u_1}\right) + \theta_2'(t)\left(-\frac{\partial \nu}{\partial u_2}\right) \tag{2.12}$$

$$= \theta_1'(t)\left(\mathsf{L}\left(-\frac{\partial f}{\partial u_1}\right)\right) + \theta_2'(t)\left(\mathsf{L}\left(-\frac{\partial f}{\partial u_2}\right)\right) \tag{2.13}$$

$$= \mathsf{L}\left(\theta_1'(t)\left(-\frac{\partial f}{\partial u_1}\right) + \theta_2'(t)\left(-\frac{\partial f}{\partial u_2}\right)\right) \tag{2.14}$$

$$= \mathsf{L}\left(\frac{d}{dt}\left[f\left(\theta(t)\right)\right]\right) = \mathsf{L}\left(\frac{d}{dt}\left[c(t)\right]\right) = \mathsf{L}\left(X\right) \tag{2.15}$$

With this, we can re-express the curvature of our curve as

$$\|c''\| = \langle X, -\frac{\partial \nu}{\partial t}\rangle = \langle X, \mathsf{L}(X)\rangle \tag{2.16}$$

which only depends on the point u and the selected direction X, not on the particular curve at all. □

In fact, we refer to this quantity as the normal curvature of the surface.

Definition 2.2.6. *The normal curvature of a surface at point u in the direction X is given by* $\kappa_\nu := \langle X, \mathsf{L}(X)\rangle$.

In fact, theorem 2.2.2 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. In other contexts (not necessary here), this quantity is referred to as the *second fundamental form* at the point $u \in U$; that is, $\mathbf{II}(X,X) := -\langle X, \mathsf{L}(X)\rangle$.

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 $\mathsf{L}(\frac{\partial f}{\partial u_i}) = -\frac{\partial \nu}{\partial u_i}$.

Definition 2.2.7 (Weingarten map)**.** *The Weingarten map (or shape operator) is the map* $\mathsf{L} : T_u f \to T_u f$ *given by* $\mathsf{L} = D\nu \circ (Df)^{-1}$.

[Fix this notational nightmare]

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 [see Kuhnel] **[TODO: real citations]**, which is certainly useful for surfaces f that are not graphs. The situation is much simpler if f is a graph–the linear transformation may be simply constructed.

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 refering to this 3-vector expanded w.r.t. a basis for $T_u f$). In matrix form, we describe this situation as

$$\begin{bmatrix} \widehat{\mathsf{L}} \end{bmatrix} \begin{bmatrix} \uparrow & \uparrow \\ \left.\frac{\partial f}{\partial u_1}\right|_{T_u f} & \left.\frac{\partial f}{\partial u_2}\right|_{T_u f} \\ \downarrow & \downarrow \end{bmatrix} = \begin{bmatrix} \uparrow & \uparrow \\ \widehat{\mathsf{L}}\left.\frac{\partial f}{\partial u_1}\right|_{T_u f} & \widehat{\mathsf{L}}\left.\frac{\partial f}{\partial u_2}\right|_{T_u f} \\ \downarrow & \downarrow \end{bmatrix} \tag{2.17}$$

$$= \begin{bmatrix} \uparrow & \uparrow \\ -\left.\frac{\partial \nu}{\partial u_1}\right|_{T_u f} & -\left.\frac{\partial \nu}{\partial u_2}\right|_{T_u f} \\ \downarrow & \downarrow \end{bmatrix} \tag{2.18}$$

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

$$\implies \begin{bmatrix} \widehat{\mathsf{L}} \end{bmatrix} \begin{bmatrix} \leftarrow \frac{\partial f}{\partial u_1} \rightarrow \\ \leftarrow \frac{\partial f}{\partial u_2} \rightarrow \end{bmatrix} \begin{bmatrix} \uparrow & \uparrow \\ \frac{\partial f}{\partial u_1} & \frac{\partial f}{\partial u_2} \\ \downarrow & \downarrow \end{bmatrix} = \begin{bmatrix} \leftarrow \frac{\partial f}{\partial u_1} \rightarrow \\ \leftarrow \frac{\partial f}{\partial u_2} \rightarrow \end{bmatrix} \begin{bmatrix} \uparrow & \uparrow \\ -\frac{\partial \nu}{\partial u_1} & -\frac{\partial \nu}{\partial u_2} \\ \downarrow & \downarrow \end{bmatrix} \tag{2.19}$$

We can simplify this greatly by defining

$$g_{ij} := \langle \frac{\partial f}{\partial u_i}, \frac{\partial f}{\partial u_j} \rangle \quad \text{and} \quad h_{ij} := \langle \frac{\partial f}{\partial u_i}, -\frac{\partial \nu}{\partial u_j} \rangle \tag{2.20}$$

so that

$$\begin{bmatrix} \widehat{\mathsf{L}} \end{bmatrix} \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \tag{2.21}$$

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

$$\widehat{\mathsf{L}} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix}^{-1} \tag{2.22}$$

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

First, we need the following lemma:

Lemma 2.2.3. *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 $\nabla_v \langle v, Av \rangle$:

$$\Big(\nabla_v\langle v,Av\rangle\Big)_i = \frac{\partial}{\partial v_i}\Big[\langle v,Av\rangle\Big] = \frac{\partial}{\partial v_i}\left[\sum_{j=1}^{n} v_j (Av)_j\right] \tag{2.23}$$

$$= \frac{\partial}{\partial v_i}\left[\sum_{j=1}^{n} v_j \sum_{k=1}^{n} a_{jk}v_k\right] \tag{2.24}$$

$$= \frac{\partial}{\partial v_i}\left[a_{ii}v_i^2 + v_i\sum_{k\neq i} a_{ik}v_k + v_i\sum_{j\neq i} a_{ji}v_j + \sum_{j\neq i}\sum_{k\neq i} v_j a_{jk}v_k\right] \tag{2.25}$$

$$= 2a_{ii}v_i + \sum_{k\neq i} a_{ik}v_k + \sum_{j\neq i} a_{ji}v_j + 0 \tag{2.26}$$

$$= 2a_{ii}v_i + 2\sum_{k\neq i} a_{ik}v_k = 2\sum_{k=1}^{n} a_{ik}v_k = 2\big(Av\big)_i \tag{2.27}$$

$$\implies \nabla_v\langle v,Av\rangle = 2Av. \tag{2.28}$$

□

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

Theorem 2.2.4 (Theorem of Olinde Rodrigues)**.** *Fixing a point $u \in U$, a direction $X \in T_u f$ minimizes the normal curvature $\kappa_v = \langle \mathsf{L}X, X\rangle$ 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:

$$\mathscr{L}(X;\lambda) := \langle \widehat{\mathsf{L}}X, X\rangle - \lambda\Big(\langle X,X\rangle - 1\Big) \tag{2.29}$$

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

$$\begin{cases} \nabla_X\langle \widehat{\mathsf{L}}X, X\rangle - \lambda\nabla_X\left(\langle X,X\rangle - 1\right) = 0 \\ \qquad\qquad\qquad\qquad \langle X,X\rangle - 1 = 0 \end{cases} \tag{2.30}$$

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

$$\begin{aligned} \nabla_X \langle \widehat{\mathsf{L}} X, X \rangle - \lambda \nabla_X \left(\langle X, X \rangle - 1 \right) &= 0 \\ 2\widehat{\mathsf{L}} X - \lambda (2X) &= 0 \\ \implies \quad \widehat{\mathsf{L}} X - \lambda X &= 0 \\ \implies \quad \widehat{\mathsf{L}} X &= \lambda X \end{aligned} \tag{2.31}$$

which implies that X is an eigenvector of $\widehat{\mathsf{L}}$ with corresponding eigenvalue λ ($X \neq 0$ from the second equation of eq. (2.30)). 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 2.2.8 (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 2.2.5. *When $f : U \to \mathbb{R}^3$ is given by $(x, y) \mapsto (x, y, h(x, y))$, the matrix representation of*

the Weingarten map ~~is exactly the Hessian matrix given in (2.1)~~ is given by

$$\widehat{\mathrm{L}} = \mathrm{Hess}(h)\tilde{G}\,, \quad \text{where} \quad \tilde{G} := \frac{1}{\sqrt{1+h_x^2+h_y^2}} \begin{bmatrix} 1+h_y^2 & -h_x h_y \\ -h_x h_y & 1+h_x^2 \end{bmatrix} \tag{2.32}$$

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 $\widehat{\mathrm{L}} \approx \mathrm{Hess}$.

Proof. First, we can (using chain rule) rewrite each component as in eq. (2.20):

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

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

$$\begin{gathered} f_x = (1,0,h_x), \quad f_y = (0,1,h_y) \\ f_{xx} = (0,0,h_{xx}), \quad f_{xy} = (0,0,h_{xy}) = f_{yx}, \quad f_{yy} = (0,0,h_{yy}) \end{gathered} \tag{2.33}$$

and we have the unit normal vector (Gauss map)

$$\nu(u_1,u_2) = \frac{\frac{\partial f}{\partial x} \times \frac{\partial f}{\partial y}}{\| \frac{\partial f}{\partial x} \times \frac{\partial f}{\partial y} \|} \tag{2.34}$$

$$= \frac{(1,0,h_x) \times (0,1,h_y)}{\| \cdots \|} \tag{2.35}$$

$$= \frac{(-h_x,-h_y,1)}{\sqrt{h_x^2+h_y^2+1}} \tag{2.36}$$

We then calculate each h_{ij} as

$$\begin{aligned} h_{11} &= \langle \frac{\partial^2 f}{\partial x^2}, \nu \rangle = \frac{h_{xx}}{\sqrt{1+h_x^2+h_y^2}} \\ h_{12} &= \langle \frac{\partial^2 f}{\partial x \partial y}, \nu \rangle = \frac{h_{xy}}{\sqrt{1+h_x^2+h_y^2}} = h_{21} \\ h_{22} &= \langle \frac{\partial^2 f}{\partial y^2}, \nu \rangle = \frac{h_{yy}}{\sqrt{1+h_x^2+h_y^2}} \end{aligned} \tag{2.37}$$

and thus the first matrix in eq. (2.22) is given by

$$[h_{ij}] = \frac{1}{\sqrt{1+h_x^2+h_y^2}}\,\mathrm{Hess}(h) \tag{2.38}$$

To calcuate the second, we use

$$\begin{aligned} g_{ij} &= \langle \frac{\partial f}{\partial u_i}, \frac{\partial f}{\partial u_j} \rangle \\ g_{11} &= \langle f_x, f_x \rangle = 1+h_x^2 \\ g_{12} &= \langle f_x, f_y \rangle = h_x h_y = g_{21} \\ g_{22} &= \langle f_y, f_y \rangle = 1+h_y^2 \end{aligned} \tag{2.39}$$

and thus

$$[g_{ij}]^{-1} = \begin{bmatrix} 1+h_x^2 & h_x h_y \\ h_x h_y & 1+h_y^2 \end{bmatrix}^{-1} = \begin{bmatrix} 1+h_y^2 & -h_x h_y \\ -h_x h_y & 1+h_x^2 \end{bmatrix} \tag{2.40}$$

Combining $[h_{ij}]$ and $[g_{ij}]^{-1}$ from eq. (2.40) and eq. (2.38) we arrive at eq. (2.32). □

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 $\mathrm{Hess}(h)$ have the same eigenvalues and eigenvectors at these points.

To make this a little more explicit, we will calculate the Weingarten map for a relatively simple graph.

The Weingarten map of a cylindrical ridge

Let f be the graph given by

$$f : \mathbb{R}^2 \to \mathbb{R}^3 \text{ by } f(x,y) = (x,y,h(x,y)), \text{ with } h(x,y) = \begin{cases} \sqrt{r^2-x^2} & -r \le x \le r \\ 0 & \text{else} \end{cases} \tag{2.41}$$

We calculate the necessary partial derivatives of f as follows:

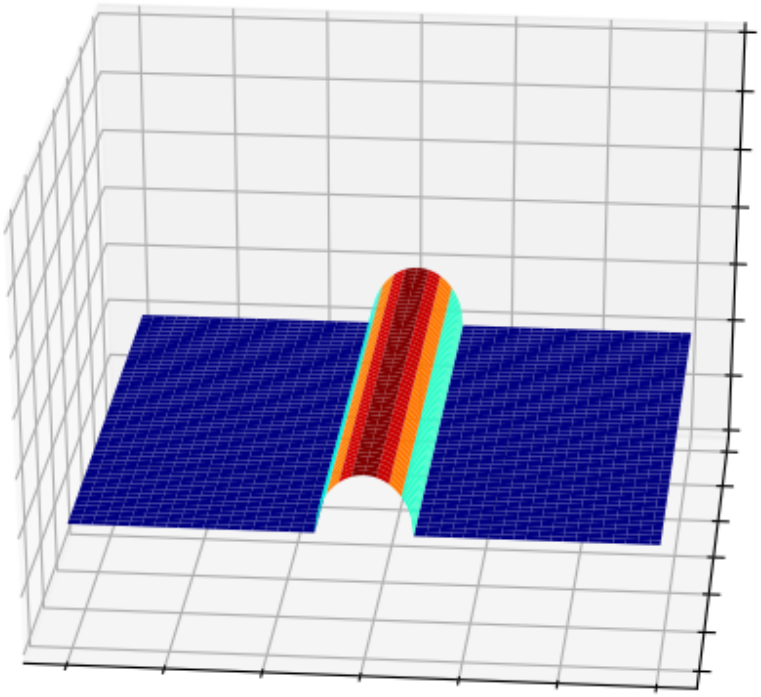


FIGURE 1. The graph of a cylindrical ridge of radius r, as given by $f(x,y)$ as given above.

$$\frac{\partial f}{\partial x} = \left(1, 0, \frac{-x}{\sqrt{r^2 - x^2}}\right) \quad , \quad \frac{\partial^2 f}{\partial x^2} = \left(0, 0, \frac{-r^2}{\left(\sqrt{r^2 - x^2}\right)^3}\right) \tag{2.42}$$

$$\frac{\partial f}{\partial y} = (0, 1, 0) \quad , \quad \frac{\partial^2 f}{\partial y^2} = \frac{\partial^2 f}{\partial x \partial y} = 0 \tag{2.43}$$

The gauss map is given by

$$\nu(x,y) = \frac{\frac{\partial f}{\partial x} \times \frac{\partial f}{\partial y}}{\| \frac{\partial f}{\partial x} \times \frac{\partial f}{\partial y} \|} = \left(\frac{x}{r},\, 0,\, \frac{\sqrt{r^2 - x^2}}{r}\right) \tag{2.44}$$

$$\implies \frac{\partial \nu}{\partial x} = \left(\frac{1}{r}\,,\, 0\,,\, \frac{-x}{r\sqrt{r^2 - x^2}}\right) \quad , \quad \frac{\partial \nu}{\partial y} = (0,0,0)\,. \tag{2.45}$$

We then calculate matrix elements of the Weingarten map's construction as given in eq. (2.38) and eq. (2.40) :

$$[h_{ij}] = \frac{1}{\sqrt{1+h_x^2+h_y^2}}\text{Hess}(h) = \frac{1}{\sqrt{1+\left(\frac{x^2}{r^2-x^2}\right)}}\begin{bmatrix}\frac{-r^2}{\sqrt{r^2-x^2}^3} & 0\\ 0 & 0\end{bmatrix} = \begin{bmatrix}\frac{-r}{r^2-x^2} & 0\\ 0 & 0\end{bmatrix} \tag{2.46}$$

$$[g_{ij}]^{-1} = \begin{bmatrix}\frac{r^2-x^2}{r^2} & 0\\ 0 & 1\end{bmatrix} \tag{2.47}$$

$$\implies \widehat{\mathrm{L}} = [h_{ij}][g_{ij}]^{-1} = \begin{bmatrix}\frac{-r}{r^2-x^2} & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}\frac{r^2-x^2}{r^2} & 0\\ 0 & 1\end{bmatrix} \tag{2.48}$$

$$= \begin{bmatrix}-\frac{1}{r} & 0\\ 0 & 0\end{bmatrix} \tag{2.49}$$

We see that $u_2 = (0,1)$ and $u_1 = (1,0)$ are eigenvectors for $\widehat{\mathrm{L}}$ 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 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.

[CLEANUP: this was dropped in here. fix the transition] Viewing the surface in $\mathbb{R}^3$, we define the Hessian *[notation here, does it need to be an operator?]* Hess of the surface L at a point (x,y) on the surface as the matrix of its second partial derivatives:

$$\text{Hess}(x,y) = \begin{bmatrix}L_{xx}(x,y) & L_{xy}(x,y)\\ L_{yx}(x,y) & L_{yy}(x,y)\end{bmatrix} \tag{2.50}$$

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

$$\text{Hess}u_i = \kappa_i u_i\,, \quad i = 1,2 \tag{2.51}$$

[CLEANUP: the eigenvectors of the Weingarten map are orthonormal. the eigenvectors of the hessian are also orthonormalizable. make the distinction!] where κ_i and u_i are known as the *principal curvatures* and *principal directions [fixthis]* of $L(x,y)$, respectively, and we label such that $|\kappa_2| \geq |\kappa_1|$. Notably, $\text{Hess}(x,y)$ 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 lemma:

Lemma 2.2.6 (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

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

[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.

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

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 $L(x,y)$.

[The following unverified (but intuitive) theorem is currently unnecessary but could be useful if I still want to implement the idea of tracking direction of eigenvectors along potential ridges, useful for filling in the ridge network. Like approximating the surface using derivatives.]

The following is an **unverified claim** (which might be useful for later): The frame varies continuously along paths in $\mathbb{R}^2$ except at points where $\text{Hess}(x,y)$ is singular. To make this explicit:

Theorem 2.2.7 (Continuity of the leading principal direction)**.** *Let* $\theta : I := [0,1] \to \mathbb{R}^2$ *be a parametrized regular curve in* $\mathbb{R}^2$ *and* $H_\theta := \text{Hess}_f \circ \theta(t)$ *be the matrix-valued function (where*

Hess_f *is the* 2×2 *Hessian of the smooth surface* f*) Let* $U : I \to \mathbb{R}^2$ *be the implicitly-defined vector valued function s.t.* $U(t)$ *is the leading eigenvector of* H_θ *(and therefore the leading principal direction of* f*). That is,*

$$H_\theta\, U(t) = \lambda U(t) \quad \textit{with} \quad \lambda = \rho(H_\theta) \tag{2.52}$$

In other words, $|\lambda| \geq |\tilde{\lambda}|$ *for any* $\tilde{\lambda} : H_\theta\, u = \tilde{\lambda} u$ *for some* $u \neq 0$*.*

Then, $U(t)$ *is continuous in* t *whenever* $H_f(t)$ *is non-singular. [Maybe fix this so that the path avoids any nonsingular points?* $U(t)$ *isn't even well-defined at such points anyway.]*

Proof. First, we show that $U(t)$ is a well-defined function at all points t where $H_f(t)$ is non-singular. **[TODO: FIX PROOF OR REMOVE]** □

[Need a transition] To we are interested in the differentiation of 2D images, we need a way to calculate the quantities of the previous section [CLEANUP: tag]. We can simply extend these ideas of differentiation to the discrete domain.

Calculating Derivatives of Discrete Images

- **[TODO: Describe taking gradient with divided difference.]**
- **[TODO: Describe the situation: derivatives should be taken on Gaussian blur (equivalent to scale space development) [Lindeberg]. That is, you can take the derivative of either the convolved image, or you can take derivatives of the Gaussian itself, *then* convolve.]**
- **[TODO: Mention that derivatives can be taken in frequency space?]**

[describe this no matter what. if you want to calculate the derivatives in frequency space, still describe this so you can compare the results.] [CLEANUP: Move scale space theory section before this.]

The Frangi Filter

Intro to Hessian-based filters

Hessian-based filters are a family of curvilinear filters that employ the Hessian and its eigenspaces to determine regions of significant curvature within an image. Several such filters exist –see Sato [13] and Lorenz[10]. These filters use information about the principal curvatures (eigenvalues of the Hessian) at each point to

Implementation Details: Convolution Speedup via FFT

[FIRST DESCRIBE HOW YOU NEED DERIVATIVES]

As described above, the actual computation of derivatives is achieved via convolution with a gaussian. In practice, this is very slow for large scales. We instead, we perform a fast Fourier transform, which offers a speedup of N^2 operations vs $\mathcal{O}(n \cdot \log_2 n)$ operations *[extend to 2D]*

[CLEANUP: Fix this part of the outline]

Fourier Transforms

Fourier Transform of a continuous 1D signal .

[start with 1D but then extend/rewrite]

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

$$f(t) = \sum_{-\infty}^{\infty} c_n e^{i\frac{2\pi n}{T}t}, \quad c_n = \frac{1}{T}\int_{-T/2}^{T/2} f(t) e^{-i\frac{2\pi n}{T}t} dt \tag{2.53}$$

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

$$F(\mu) := \mathscr{F}\{f(t)\} = \int_{-\infty}^{\infty} f(t) e^{i2\pi\mu} dt \tag{2.54}$$

It can be shown *[justify?]* that an inverse transform will then recover our original signal:

$$f(t) = \mathscr{F}^{-1}\{F(\mu)\} = \int_{-\infty}^{\infty} F(\mu) e^{i2\pi\mu t} dt \tag{2.55}$$

Together, eq. (2.54) and eq. (2.55) 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. We frame the situation as follows: A continuous function $f(t)$ *[right now, with domain $\mathbb{R}$]* is represented as the sampled function $\tilde{f}(t)$ by multiplying it by a sampling function (also referred to in DIP-GW as an impulse function), an infinite series of discrete impulses with equal spacing ΔT:

$$s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} \delta[t - n\Delta T] \,, \quad \delta[t] = \begin{cases} 1 \,, & t = 0 \\ 0 \,, & t \neq 0 \end{cases} \tag{2.56}$$

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

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

$$\tilde{f}(t) = f(t) s_{\Delta T}(t) \tag{2.57}$$

From this (and actually the convolution theorem) we can calculate $\tilde{F}(t)$. **[TODO: Derive this. There's some groundwork here to copy between 213-220. We don't need the sampling theorem, since our signal is given as-is... i think]**.

1D Discrete Fourier Transform (DFT) .

Given the discrete signal $\tilde{f}$, we construct the transform $\tilde{F}(\mu) = \mathscr{F}\{\tilde{f}(t)\}$. by simply transforming the definition eq. (2.57).

[supress derivation but do explain f_n notation / cleanup]

$$\tilde{F}(\mu) = \sum_{n=-\infty}^{\infty} f_n e^{-i2\pi\mu n\Delta T} \,, \quad f_n = \tilde{f}(n) = f(n\Delta T) \tag{2.58}$$

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

- **[TODO: find image processing papers that find hessian from FFT / who uses this?]**
- **[TODO: with above: downsides?]**
- **[TODO: side by side comparison in a toy example and/or a real problem?]**

2D Discrete Fourier Transform Convolution Theorem . *[the following was adapted in a large part from DFT: an owner's manual. cite? DIP-DW just proves the continous version (in 1D) and then asserts that it works for discrete variables too.]*

[CLEANUP: get consistent notation–either have the discrete signals be notated as $\tilde{f}(x,y)$ or $f[x,y]$ or instead comment that it's understood]

Theorem 2.5.1 (2D DFT Convolution Theorem)**.** *[develop the 2-D DFT from Sec. 4.3 4.4 from DIP-GW (see p235).] Given two discrete functions are sequences with the same length* ***[TODO: don't gloss over this]*** *[If they're not actually the same length, DIP-GW suggests to make the final length at least $P = A + C - 1$ and $Q = B + D - 1$ in the case that the sizes are $A \times B$ and $C \times D$ for $f(x,y)$ and $h(x,y)$ respectively. Not sure if that matters.], that is: $f(x,y)$ and $h(x,y)$ for integers $0 < x < M$ and $0 < y < N$, we can take the discrete fourier transform (DFT) of each:*

$$F(u,v) := \mathcal{D}\{f(x,y)\} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-2\pi i\left(\frac{ux}{M}+\frac{vy}{N}\right)} \tag{2.59}$$

$$H(u,v) := \mathcal{D}\{h(x,y)\} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} h(x,y)e^{-2\pi i\left(\frac{ux}{M}+\frac{vy}{N}\right)} \tag{2.60}$$

and given the convolution of the two functions

$$(f \star h)(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)h(x-m,y-n) \tag{2.61}$$

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

$$(f \star h)(x,y) = \mathcal{D}^{-1}\{MN \cdot F(u,v)H(u,v)\} \tag{2.62}$$

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

Proof.

$$(f \star h)(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)h(x-m,y-n) \tag{2.63}$$

$$= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left(\sum_{p=0}^{M-1}\sum_{q=0}^{N-1} F(p,q)e^{2\pi i\left(\frac{mp}{M}+\frac{nq}{N}\right)}\right)\left(\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} H(u,v)e^{2\pi i\left(\frac{u(x-m)}{M}+\frac{v(y-n)}{N}\right)}\right) \tag{2.64}$$

$$= \left(\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} H(u,v)e^{2\pi i\left(\frac{ux}{M}+\frac{vy}{N}\right)}\right)\left(\sum_{p=0}^{M-1}\sum_{q=0}^{N-1} F(p,q)\left(\sum_{m=0}^{M-1} e^{2\pi i\left(\frac{m(p-u)}{M}\right)}\right)\left(\sum_{n=0}^{N-1} e^{2\pi i\left(\frac{n(q-v)}{N}\right)}\right)\right) \tag{2.65}$$

$$= \left(\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} H(u,v)e^{2\pi i\left(\frac{ux}{M}+\frac{vy}{N}\right)}\right)\left(\sum_{p=0}^{M-1}\sum_{q=0}^{N-1} F(p,q)\left(M\cdot\hat{\delta}_M(p-u)\right)\left(N\cdot\hat{\delta}_M(q-v)\right)\right) \tag{2.66}$$

$$= \left(\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} H(u,v)e^{2\pi i\left(\frac{ux}{M}+\frac{vy}{N}\right)}\right)\cdot MNF(u,v) \tag{2.67}$$

$$= MN\cdot\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)H(u,v)e^{2\pi i\left(\frac{ux}{M}+\frac{vy}{N}\right)} \tag{2.68}$$

$$= MN\cdot\mathcal{D}^{-1}\{FH\} \tag{2.69}$$

where

$$\hat{\delta}_N(k) = \begin{cases} 1 & \text{when } k = 0 \mod N \\ 0 & \text{else} \end{cases} \tag{2.70}$$

□

Above, we make use of the following lemma *[add this before DFT convolution theorem and embed the definition of $\hat{\delta}_N$ inside]*

Lemma 2.5.2. *Let j and k be integers and let N be a positive integer. Then*

$$\sum_{n=0}^{N-1} e^{2\pi i\left(\frac{n(j-k)}{N}\right)} = N\cdot\hat{\delta}_N(j-k) \tag{2.71}$$

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

$$z^N - 1 \;=\; (z-1)\left(z^{n-1} + \cdots + z + 1\right) \;=\; (z-1)\sum_{n=0}^{N-1} z^n. \tag{2.72}$$

thus giving us

$$0 = \left(e^{2\pi i(j-k)/N} - 1\right)\sum_{n=0}^{N-1} e^{2\pi in(j-k)/N} \tag{2.73}$$

To prove the claim in eq. (2.71), we consider two cases: First, if $j-k$ is a multiple of N, we of course have $e^{2\pi in(j-k)/N} = \left(e^{2\pi i}\right)^{n(j-k)/N} = 1$ and thus the left side of eq. (2.71) reduces to

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

.

In the case that $j-k$ is *not* a multiple of N, we refer to eq. (2.73). The first factor is not zero since, $\left(e^{2\pi i(j-k)/N}\right) \neq 1$ (simply since $(j-k)/N$ is not an integer), and thus it must be that the second factor is 0:

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

We can combine these two cases by invoking the definition of eq. (2.70), giving us the result. □

FFT . *[use DIP-GW p298]* As noted, the above result applies to the Discrete Fourier Transform. As noted, we actually achieve a convolution speedup using a Fast Fourier Transform (FFT) instead. We follow the developments of DIP-GW *[should I? or just link? or fix notation? or do for 2D at the same time]*. 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:

$$F(\mu,\nu) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-i2\pi(\mu x/M+\nu y/N)} \tag{2.74}$$

$$= \sum_{x=0}^{M-1} e^{-i2\pi\mu x/M}\left[\sum_{y=0}^{N-1} f(x,y)e^{-i2\pi\nu y/N}\right] \tag{2.75}$$

$$= \sum_{x=0}^{M-1} e^{-i2\pi\mu x/M}\,\mathscr{F}_x\{f(x,y)\} \tag{2.76}$$

$$= \mathscr{F}_y\{\mathscr{F}_x\{f(x,y)\}\} \tag{2.77}$$

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 $x \in 0..M-1$. 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 $(MN)^2$ *[rectify this, is it really caused by the number of u,v points that need calculating?]*. (Note: the values of $e^{-i2\pi m/n}$ can be provided by a lookup table rather than ad-hoc calculation *[source?]*.

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.

[we follow DIP-GW variable conventions, which I think are dumb]

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

$$F(u) = \sum_{x=0}^{M-1} f(x)W_M^{ux}, \quad \text{where} \quad W_m := e^{-i2\pi/m} \tag{2.78}$$

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. (2.78) and split it into odd and even entries in the summation

$$F(u) = \sum_{x=0}^{2K-1} f(x) W_{2K}^{ux} \tag{2.79}$$

$$= \sum_{x=0}^{K-1} f(2x) W_{2K}^{u(2x)} + \sum_{x=0}^{K-1} f(2x+1) W_{2K}^{u(2x+1)} \tag{2.80}$$

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

[this fixes an issue in DIP-GW where the identities were provided in terms of M instead of arbitrary m, where the proof uses the results for some value other than M anyway]

$$W_{(2m)}^{(2n)} = e^{\frac{-i2\pi(2m)}{2m}} = e^{\frac{-i2\pi m}{n}} = W_m^n \tag{2.81}$$

$$W_m^{(u+m)x} = e^{\frac{-i2\pi(u+m)x}{m}} = e^{\frac{-i2\pi unx}{m}} e^{\frac{-i2\pi mx}{m}} = e^{\frac{-i2\pi ux}{m}}(1) = W_m^{ux} \tag{2.82}$$

$$W_{2m}^{(u+m)} = e^{\frac{-i2\pi(u+m)}{2m}} = e^{\frac{-i2\pi ux}{2m}} e^{-i\pi} = W_{2m}^u e^{-i\pi} = -W_{2m}^u \tag{2.83}$$

Thus we can rewrite eq. (2.80) as

$$F(u) = \sum_{x=0}^{K-1} f(2x) W_{2K}^{2ux} + \sum_{x=0}^{K-1} f(2x+1) W_{2K}^{2ux} W_{2K}^u \tag{2.84}$$

$$\implies \quad F(u) = \left(\sum_{x=0}^{K-1} f(2x) W_K^{ux} \right) + \left(\sum_{x=0}^{K-1} f(2x+1) W_K^{ux} \right) W_{2K}^u \tag{2.85}$$

The major advance comes via using the identities eq. (2.81) [CLEANUP: fix multitag] to consider the Fourier transform K frequencies later *[wording?]*:

$$F(u+K) = \left(\sum_{x=0}^{K-1} f(2x) W_K^{(u+K)x} \right) + \left(\sum_{x=0}^{K-1} f(2x+1) W_K^{(u+K)x} \right) W_{2K}^{(u+K)} \tag{2.86}$$

$$\implies F(u+K) = \left(\sum_{x=0}^{K-1} f(2x) W_K^{ux} \right) - \left(\sum_{x=0}^{K-1} f(2x+1) W_K^{ux} \right) W_K^u \tag{2.87}$$

Comparing eq. (2.85) and eq. (2.87), 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:

$$\mathscr{D}_u\{f_{\text{even}}(t)\} := \sum_{x=0}^{K-1} f(2x)W_K^{ux} \tag{2.88}$$

$$\mathscr{D}_u\{f_{\text{odd}}(t)\} := \sum_{x=0}^{K-1} f(2x+1)W_K^{ux} \tag{2.89}$$

If we're calculating an M point transform **[TODO: vocabulary also how many frequencies do we calculate? same as # samples? what do we need?]** (i.e. we're wishing to calculate $F(1),...,F(M)$), once we've calculated the first K discrete frequencies (i.e. $F(1),..,F(K)$) we may simply reuse the two values we've calculated in eq. (2.88) 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 $M\log_2 M$. **[TODO: finish this]**.

Of course, since eq. (2.88) 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

[CLEANUP: cite previous section or move section here.]

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 **[TODO: probably]** that you just do the part that's a power of 2 and then do the rest manually or pick a different power. See **[TODO: find out how fft-pack does this]**

The inverse DFT is actually a DFT of a modified function, see **[TODO: cite DIP-GW here or show that]**. **[TODO: check how fftpack does this]**

Linear Scale Space Theory

[this is all as cited in GSST book (see starting in sec 6.3.1). as mentioned there, this is analogous to a discrete development (lindeberg 1990 lindeberg 1991 lindeberg 1994c lindeberg 1994e)]

Koenderink showed/asserted 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*. They showed in particular that any such family must satisfy the heat equation

$$\Delta K(x,y,\sigma) = K_\sigma(x,y,\sigma) \text{ for } \sigma \geq 0 \text{ such that } K(x,y,0) = u_0(x,y). \tag{2.90}$$

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. There is a long list of desired properties quoted from gsst. goal is to try to find a minimal subset of axioms and show that other desired properties follow.

Axioms

To make matters manageable, [PEOPLE] have taken the approach of requiring the one-parameter family of scaled images to be generated by an operation.

Linear-shift invariance

This requires that the generator is convolution by some kernel. Show or proclaim. Shift-invariant means that no position in the original signal is favored (makes sense, this should apply to any image.)

Rotational invariance . Similar to above, no favoritism to directionality.

Semigroup property . They stack (no preferred scale)

Causality condition . No "spurious resolution" can be generated as parameter increases. *[FIX NOTATION]* If $K(x_0, y_0; z_0)$ is a local maximum (in space) (i.e. $\nabla_{x,y} K(x_0, y_0; z_0) = 0$, $\Delta_{x,y} K(x_0, y_0; z_0) < 0$), then an increase in scale can only weaken this peak,

i.e.

$$z_0 > z_1 \implies K(x_0, y_0; z_0) \geq K(x_0, y_0; z_1) \tag{2.91}$$

Similarly, if $K(x_0, y_0; z_0)$ is a local minimum (in space), (i.e. $\nabla_{x,y} K(x_0, y_0; z_0) = 0$, $\Delta_{x,y} K(x_0, y_0; z_0) > 0$), then an increase in scale cannot make such a valley more profound, i.e.

$$z_0 > z_1 \implies K(x_0, y_0; z_0) \leq K(x_0, y_0; z_1) \tag{2.92}$$

This implies that no image feature is sharpened by an decrease and resolution–the only result is a flattening out of the image as scale parameter z tends to infinity.s

Linearity of generator . "Linearity implies that all-scale space properties valid for the original signal will transfer to its derivatives. Hence, there is no commitment to certain aspects of image structure, such as the zero-order representation, or its first- or second-order derivatives." *[this is dubious, as mentioned by JvB]*

Necessity of convolution

Where is this.

Uniqueness of the Gaussian Kernel

Proved *[however artificially]* by Koenderink [1984] and Babaud [1986] that such an axiomatic foundation requires that the family is generated by convolution with a kernel. The remainder of the development is to the determine that the kernel is exactly the Gaussian.

[TODO: TODO: JUST EVEN OUT THE ABOVE. FIND SOMETHING TO JUSTIFY CONVOLUTION. CITATIONS FOR LATER FORMULATIONS. BUILD UP HEAT EQUATION (HOWEVER RIGOROUSLY YOU CAN). SHOW NECESSITY OF GAUSSIAN (OR DON'T).]

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 (koenderink paper?)

$G_\sigma \star u_0$ solves the heat equation

given u_0 as a continuous image (unscaled), we construct PDE with this as a boundary condition.

$$u : \mathbb{R}^2 \supset \Omega \to \mathbb{R} \text{ with } u(\boldsymbol{x},t) : \begin{cases} \frac{\partial u}{\partial t}(\boldsymbol{x},t) = \Delta u(\boldsymbol{x},t) \quad , t \geq 0 \\ u(\boldsymbol{x},0) = u_0(\boldsymbol{x}) \end{cases} \tag{2.93}$$

We show that

$$u(\boldsymbol{x},t) = \left(G_{\sqrt{2t}} \star u_0\right)(\boldsymbol{x}) \tag{2.94}$$

solves (the above tagged equation), where

$$G_\sigma := \frac{1}{2\pi\sigma^2} e^{\left(-|x|^2/(2\sigma^2)\right)}$$

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

Lemma 2.6.1. *Derivative of a convolution is the way that it is (obviously rewrite this).*

Proof. For a single variable,

$$\frac{\partial}{\partial\alpha}\left[f(\alpha) \star g(\alpha)\right] = \frac{\partial}{\partial\alpha}\left[\int f(t)g(\alpha - t)dt\right] \tag{2.95}$$

$$= \int f(t)\frac{\partial}{\partial\alpha}\left[g(\alpha - t)\right]dt \tag{2.96}$$

$$= \int f(t)\left(\frac{\partial g}{\partial\alpha}\right)g(\alpha - t)dt \tag{2.97}$$

$$= f(\alpha) \star g'(\alpha) \tag{2.98}$$

By symmetry of convolution we can also conclude

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

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

$$\Delta(f \star g) = \Delta(f) \star g = f \star \Delta(g) \tag{2.99}$$

□

Theorem 2.6.2. *$u(\boldsymbol{x},t) = \left(G_{\sqrt{2t}} \star u_0\right)(\boldsymbol{x})$ solves the heat equation.*

Proof. We focus on the particular kernel

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

Then

$$\begin{aligned}
\frac{\partial u}{\partial t}(\boldsymbol{x},t) &= \frac{\partial}{\partial t}\left(G_{\sqrt{2t}}(\boldsymbol{x},t) \star u_0(\boldsymbol{x})\right) && (2.100)\\
&= \frac{\partial}{\partial t}\left(G_{\sqrt{2t}}(\boldsymbol{x},t)\right) \star u_0(\boldsymbol{x}) && (2.101)\\
&= \frac{\partial}{\partial t}\left(\frac{1}{4\pi t} e^{\left(-|x|^2/(4t)\right)}\right) \star u_0(\boldsymbol{x}) && (2.102)\\
&= \left[-\frac{1}{4\pi t^2} e^{\left(-|x|^2/(4t)\right)} + \frac{1}{4\pi t}\left(\frac{-|x|^2}{4t^2}\right) e^{-|x|^2/(4t)}\right] \star u_0(\boldsymbol{x}) && (2.103)\\
&= -\frac{1}{4t^2}\left(e^{\left(-|x|^2/(4t)\right)} + |\boldsymbol{x}|^2 G_{\sqrt{2t}}(\boldsymbol{x},t)\right) \star u_0(\boldsymbol{x}) && (2.104)
\end{aligned}$$

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:

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

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

$$
\Delta G_\sigma(x,y) = \frac{\partial^2}{\partial x^2}\left(G_\sigma\right)+\frac{\partial^2}{\partial y^2}\left(G_\sigma\right) = \frac{1}{\sigma^2}G_\sigma(x,y)\left[\frac{x^2+y^2}{\sigma^2}-2\right] \tag{2.105}
$$

Then, given lemma 2.6.1, we conclude

$$
\Delta\left[G_\sigma(x,y)\star u_0(x,y)\right] = \left(\frac{1}{\sigma^2}G_\sigma(x,y)\left[\frac{x^2+y^2}{\sigma^2}-2\right]\right)\star u_0(x,y) \tag{2.106}
$$

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

$$
\begin{aligned}
\Delta\left[G_{\sqrt{2t}}(x,y)\star u_0(x,y)\right] &= \left(\frac{1}{2t}G_{\sqrt{2t}}(x,y)\left[\frac{x^2+y^2}{2t}-2\right]\right)\star u_0(x,y) &(2.107)\\
&= \left(G_{\sqrt{2t}}(x,y)\left[\frac{x^2+y^2}{4t^2}-\frac{1}{t}\right]\right)\star u_0(x,y) &(2.108)
\end{aligned}
$$

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

$$\frac{\partial}{\partial t}\left[G_{\sigma(t)}(x,y)\star u_0(x,y)\right] = \frac{\partial}{\partial t}\left[G_{\sigma(t)}(x,y)\right]\star u_0(x,y) \tag{2.109}$$

$$= \frac{\partial}{\partial t}\left[G_{\sqrt{2t}}(x,y)\right]\star u_0(x,y) \tag{2.110}$$

$$= \frac{\partial}{\partial t}\left[\frac{1}{4\pi t}\exp\left(-\frac{x^2+y^2}{4t}\right)\right]\star u_0(x,y) \tag{2.111}$$

$$= \left[-\frac{1}{4\pi t^2}\exp\left(-\frac{x^2+y^2}{4t}\right)+\frac{1}{4\pi t}\left(\frac{x^2+y^2}{4t^2}\exp\left(-\frac{x^2+y^2}{4t}\right)\right)\right]\star u_0(x,y) \tag{2.112}$$

$$= \left(G_{\sqrt{2t}}(x,y)\left[\frac{x^2+y^2}{4t^2}-\frac{1}{t}\right]\right)\star u_0(x,y) \tag{2.113}$$

Combining these results, we find that

$$\frac{\partial}{\partial t}\left[G_{\sqrt{2t}}\star u_0\right] = \Delta\left[G_{\sqrt{2t}}\star u_0\right] \tag{2.114}$$

as desired. □

Alternative formulations of Scale Space Theory

Mention discrete treatments, mention relaxation of some of the dubious axioms.

Morphology

Methods of merging multiscale methods.

Overview of Frangi vesselness measure

[CLEANUP: move to next section? or cover any necessary math here]

The Frangi filter [1] is a widely used *[citation needed]* Hessian-based filter used to detect curvilinear structures in images. It was orignally developed for images such as MRIs and it excels in that context. It essentially relies on measuring eigenvalues of the Hessian $\text{Hess}_\sigma(x,y)$ at some particular scale σ at each point (x,y) in the image and comparing their magnitudes.

The procedure for a single scale in a 2D image is as follows:

[CLEANUP: redefine this in a sensible way so you can easily talk about the point in the domain of the image, etc.] Let λ_1, λ_2 be the two eigenvalues of the Hessian of the image at point (x_0, y_0), ordered such that $|\lambda_1| \leq |\lambda_2|$, and define the Frangi vesselness measure at scale σ as:

$$V_\sigma(x_0, y_0) = \begin{cases} 0 & \text{if} \quad \lambda_2 > 0 \\ \exp\left(-\frac{A^2}{2\beta^2}\right)\left(1 - \exp(-\frac{S^2}{2c^2})\right) & \text{else} \end{cases} \tag{2.115}$$

where

$$A := |\lambda_1/\lambda_2| \quad \text{and} \quad S := \sqrt{\lambda_1^2 + \lambda_2^2} \tag{2.116}$$

and β and c are tuning parameters. Before we discuss appropriate values for β and c, we first seek to highlight the significance of **??**, and in particular, the ratios defined in eq. (2.116). A and S are known as the anisotropy measure and structureness measure, respectively. [CLEANUP: define these earlier than the vesselness measure so that the definitions are easier to find].

Anisotropy

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.

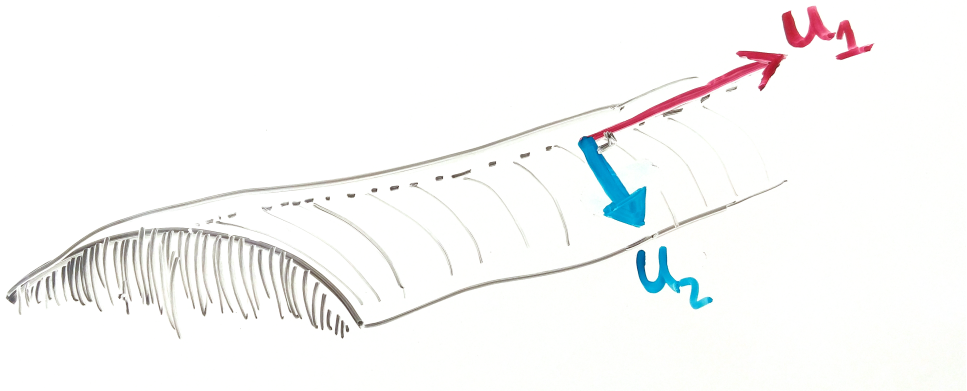


FIGURE 2. The principal eigenvectors of the image L at a ridge like structure (arrows not to scale). $\lambda_1 \approx 0$ and λ_2 is large and negative. [TODO: fix drawing so u_1 is more to scale]

In section 2.8.1, this situation is demonstrated. Here, u_1, u_2 is 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).

Of course, if the the ridge is perfectly circular along its cross section (as was in section 2.2.3, it is of course apparent that λ_2 would be the same value at any place along the ridge (not just at its crest [CLEANUP: fix terminology]), and λ_1 would likewise be 0 at any such point. Thus, the anisotropy measure will not necessarily be a maximum at the crest of the ridge. 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 [CLEANUP: define structureness first]) will not in general identify the crests of a ridgelike structure–it only will highlight that such a pixel is on a ridgelike structure at all.

Similiarly, the vessel we're trying 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 we seek to identify. (either small parts where it goes up (so that u_1 is not exactly zero **[TODO: and we could fix this by checking that it's continuousish?]**, there may be small divots where the thing seems to go up a bit etc etc etc or sampling error cos it's a fucking picture etcetc **[TODO: flesh this out]**.

[CLEANUP: move elsewhere] Importantly, this formulation does allow that λ_1 is not necessarily approximately 0, just that the curvature in the downward direction is much more significant.

Also the crest could be really flat, in which case both are around zero. again, this is a scaling issue.

The fix to this is to look at scale and make sure we're dealing with an appropriate scale.

Also the fix is to find the ridge using a separate technique.

Also look at where the stitches are (discontinuities in u_2, but u_1 still 0?)

Structureness measure

There is another concern with using the pure ratio $S := |\lambda_1/\lambda_2|$ as an identifying feature of ridgelike structures: 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 quite simple fix to differentiate these points is to introduce a "structureness" measure to insure that there is in fact some 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 [CLEANUP: show?].

[CLEANUP: note somewhere (maybe in a section called 'tuning parameters' how important this one is]

The Frangi vesselness filter: Basic Anatomy

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. (2.115) 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.

We have an added case present in eq. (2.115), simply to ensure 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.

[TODO: can you plot this theoretically to show when V is large? Good parameters? Like show multiple peaks of $\exp(\dots)$ and $(1-\exp(\dots))$ arranged so that they only coincide in 'good' regions and are rapidly diminishing overlap elsewhere?]

The Frangi vesselness filter: Choosing parameters β and c

The parameters β and c are meant to scale so that the peaks of exp and $1-exp$ 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 c 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. **[TODO: find your proof/add proof]**. For β we would say that a good intermediate point is 0.5. (thus $2\beta^2 = 1/2$). **[TODO: analyze/show an example of what this does in the context of particular values of λ_1, λ_2 in terms of V output]**.

As we will show later, choosing c freely 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 **[TODO: explain better or make a picture.]** then β should be smaller. If this is not the case, a stronger filter can be created by requiring A to be much smaller.

A multiscale approach

As identified earlier [CLEANUP: cite when], scale is crucial in correctly identifying ridgelike structures. We wish to find regions that would receive a high vesselness score at any range, and consider them all together. Frangi [1] approached this problem by simply aggregating vesselness measure over all scales:

$$V(x_0, y_0) = \max_{\sigma \in \Sigma} V_\sigma(x_0, y_0) \tag{2.117}$$

where $\Sigma := \{\sigma_0, \sigma_1, \cdots, \sigma_N\}$ is a range of scale spaces 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:

$$\mathsf{V}_\Sigma := [V(x,y)]_{\substack{0 \le x < M \\ 0 \le y < N}} \tag{2.118}$$

Whereas Frangi [1] left it relatively open-ended, if we wish to be final about the whole matter, we can ultimately say whether or not a pixel does in fact corresponds to a curvilinear structure or not. There are multiple methods of doing so. A naive yet enticing approach is to simply threshold past some certain point. The resulting matrix can be given in terms of either eq. (2.117) or eq. (2.118) [CLEANUP: fix notation in all of this]

$$V_{\Sigma,\alpha}(x,y) = \begin{cases} 1 & \text{if} \quad V(x,y) \ge \alpha \\ 0 & \text{else} \end{cases}, \quad \alpha > 0 fixed. \tag{2.119}$$

[TODO: Describe an alternative approach of stitching.]

[CLEANUP: this goes somewhere else]

methods of calculating 2D Hessian

Note the following 6 methods should all theoretically be equivalent:

- convolve discrete image with a gaussian kernel, then take derivatives (no FFT). This is the "classical approach"
- Take derivatives of gaussian kernel, then convolve with the image/signal (again, no FFT)(due to [CLEANUP: cite theorem])

- FFT image and FFT gaussian kernel then convolve in freq. space, then IFFT, then take derivatives in xy-space (my implementation)
- take derivatives of gaussian kernel, then FFT it, FFT the image, then colvolve in frequency space, then IFFT (slower due to size of large kernels)
- FFT image, then multiply by theoretical gaussian kernel in freq. space, then IFFT, then take derivatives in xy-space.
- FFT image, then multiply by theoretical gaussian kernel in freq space, then take derivatives in freq space, then IFFT.

  [TODO: big implementation note: you want to show equivalence of these, but the problem is that around the boundary the issue is annoying. can you fix this?]

CHAPTER 3

IMPLEMENTATIONS

[This chapter shows how things described within the research protocol are performed. By separating it out, I can focus on things like verifying accuracy / comparisons / demos / pseudocode without cluttering up the discussion of the actual methodologies of the next chapter. That way parameters choices, etc. can be more clearly highlighted. However, this section is apt place to discuss how varying parameters influences whatever methods are being used.]

Calculating the Hessian

There are 6 ways to do this in theory. [CLEANUP: cite these]

To compare, standard gaussian calculation is implemented by `scipy.ndimage.filters.gaussian_filter`.

[CLEANUP: figure out what how to cite this, according to `https://www.scipy.org/citing.html`]

Scale Range detection

Find smallest and largest thing . . .

Morphology: Plate Eroding

. . .

Morphology/Skeletonization

. . .

CHAPTER 4

RESEARCH PROTOCOL

[List all decisions you make. Be very explicit. Current plan: Go into depth for certain portions in the implementations chapter.]

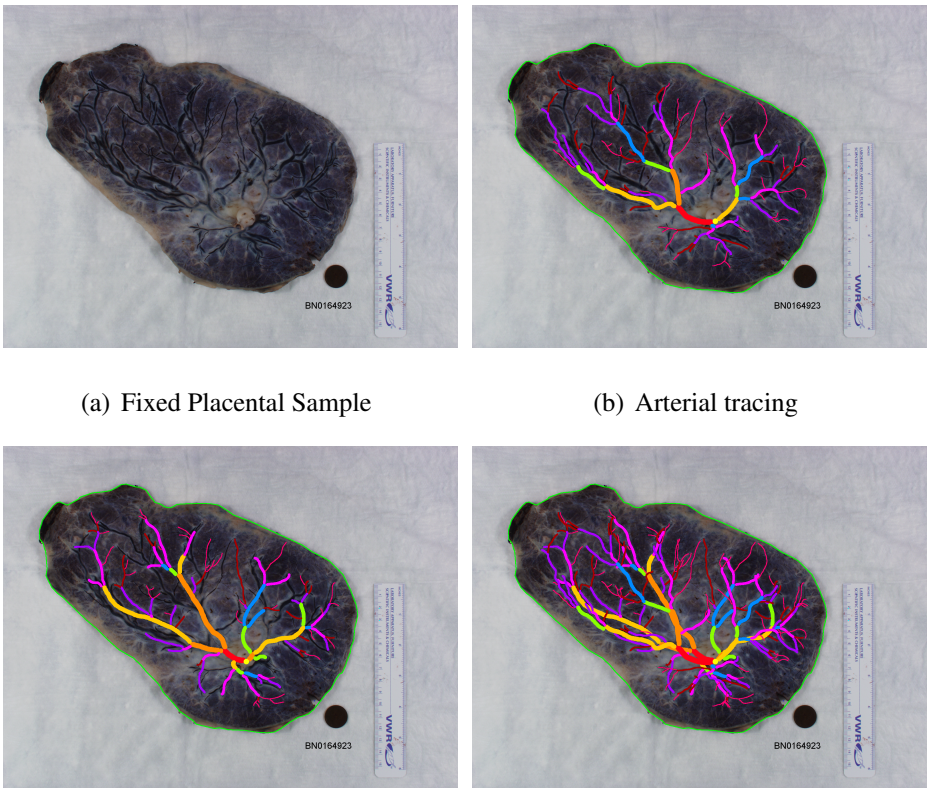
Samples / Image Domain

We ultimately perform a PCSVN extraction on a set of 174 color placental images from a private database called NCS (from NYMH?). These are project files in GIMP which contain multiple layers. The layers together give a hand tracing of the vascular network and perimeter. A sample of overlaid layers in a representative sample is given in section 4.1

In fig. 3(a), a cleaned, fixed placenta is shown. A detailed description of this procedure is given in **[TODO: some reference]**. fig. 3(b) and fig. 3(c) are both hand traces of the PCSVN, with a layer for each the arteries and veins. In our particular use case, there is no need to consider them separately, so we simply consider them together, as in fig. 3(d). The coloration is meant to indicate the diameter of each vessel. There is also a cord insertion point notated, as well as the perimeter of the placental plate. These are hand-traced and rather labor intensive. A closer look at many of the samples often reveals some subjectivity in the tracings (often it's hard to see where the vein is, vascular networks are obscured, etc.)

For our procedure, we simply operate on the placental sample itself, without any understanding of its provided tracing except for comparing the strength of our algorithm. Of course, our goal is to develop an algorithm that can produce a "ground truth" tracing such as in fig. 3(d) or fig. 4(d) without any intervention.

For our purposes however, we will use the provided placental perimeter (shown in green in section 4.1. In developing a fully automated algorithm, it would be relatively straightforward to



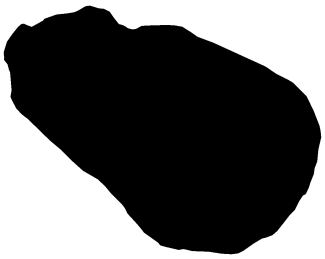
(a) Fixed Placental Sample

(b) Arterial tracing

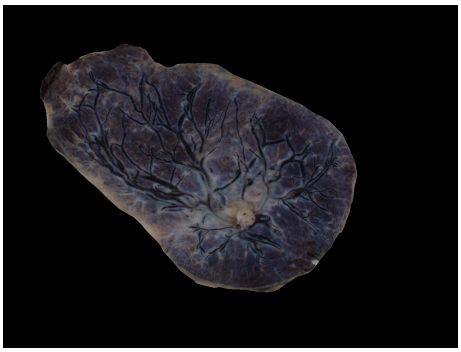
(c) Venous tracing

(d) Total Vascular Network

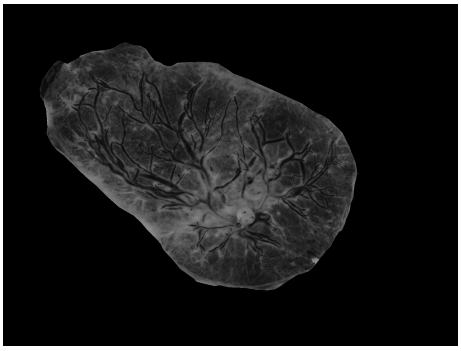
FIGURE 3. A representative placental NCS sample with vascular tracing



(a) Background Mask



(b) Sample with BG removed



(c) Grayscale

(d) Trace / “Ground Truth”

FIGURE 4. Preprocessed files from an NCS sample

obtain this boundary ourselves using an Active Contour Model **[TODO: REF]** or perhaps even any edge finding algorithm followed by morphological / watershedding as in **[TODO: REF]**.

To build a sample suitable for use in our algorithm from section 4.1 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.

These procedures are performed automatically on the 174 image 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. **[TODO: put a link here]** (There are actually 201 images but 27 of them have mislabeled layers and were not autoprocessed correctly)

Image Preprocessing

As a point of technicality, the grayscale image in fig. 4(c) is not actually produced directly by the extractor plug-in, but created when the 3 channel RGB image fig. 4(b) 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 dimension input [1], 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 [3], or “luminance” transform:

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

‘ 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 of some interest to create an algorithm that, say, calculates a

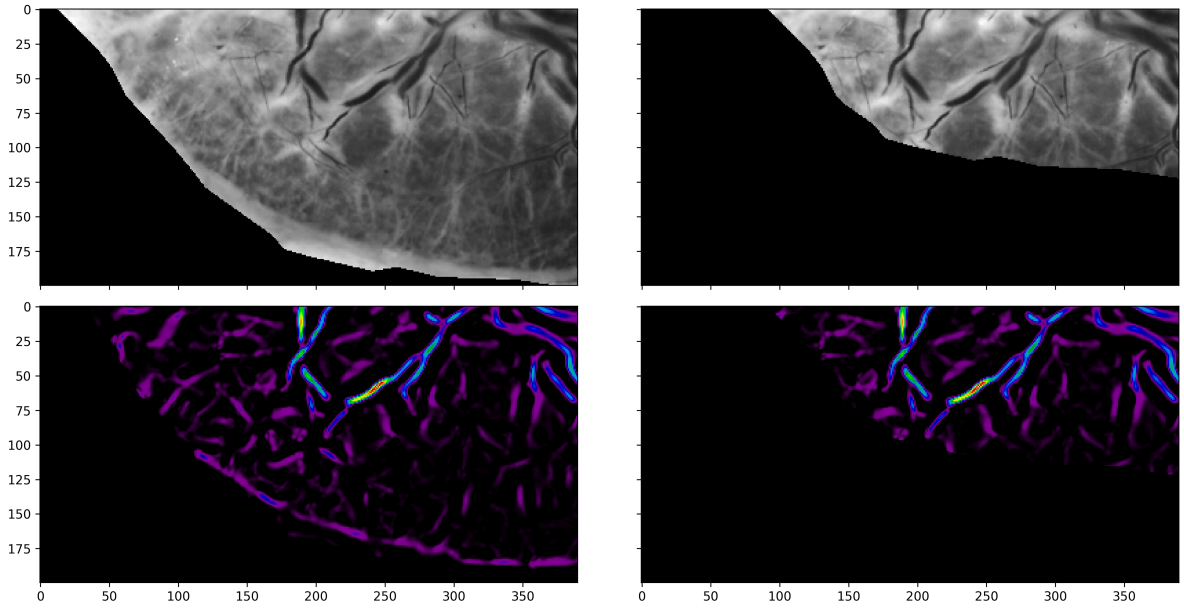


FIGURE 5. Demonstration of boundary dilation

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 simply exclude affected areas from consideration, and zero unwanted background pixels to speed up computation. This excluding function, `plate_morphology.dilate_plate`, ultimately relies on two functions provided by the Python library `scikit-image` [4]. The first, `skimage.segmentation.find_boundaries()`, takes the mask input (such as fig. 4(a)) and calculates where differences in a morphological erosion and dilation occur. That boundary itself is then dilated by the desired factor. The second is a "sparse" implementation of binary dilation that is particularly efficient for our problem. An array of indices of the image where the

4.2 doesn't really show what I want it to, but this is what it would look like. Repeat with a smaller border. Maybe the issue doesn't occur with these as much? In the image above, $\sigma = 3$ and border radius is 80 and all it does is get rid of the stupid natural boundary, not a weird frangi response. Which is important in itself, but I was having an error just between the edge of the image.

The code for the above can by found by running `plate_morphology.py` as a top-level script (that is, within the "`if __name__ == __main__`" block of the file).

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 $\Sigma := \{\sigma_1, \sigma_2, \ldots, \sigma_N\}$.

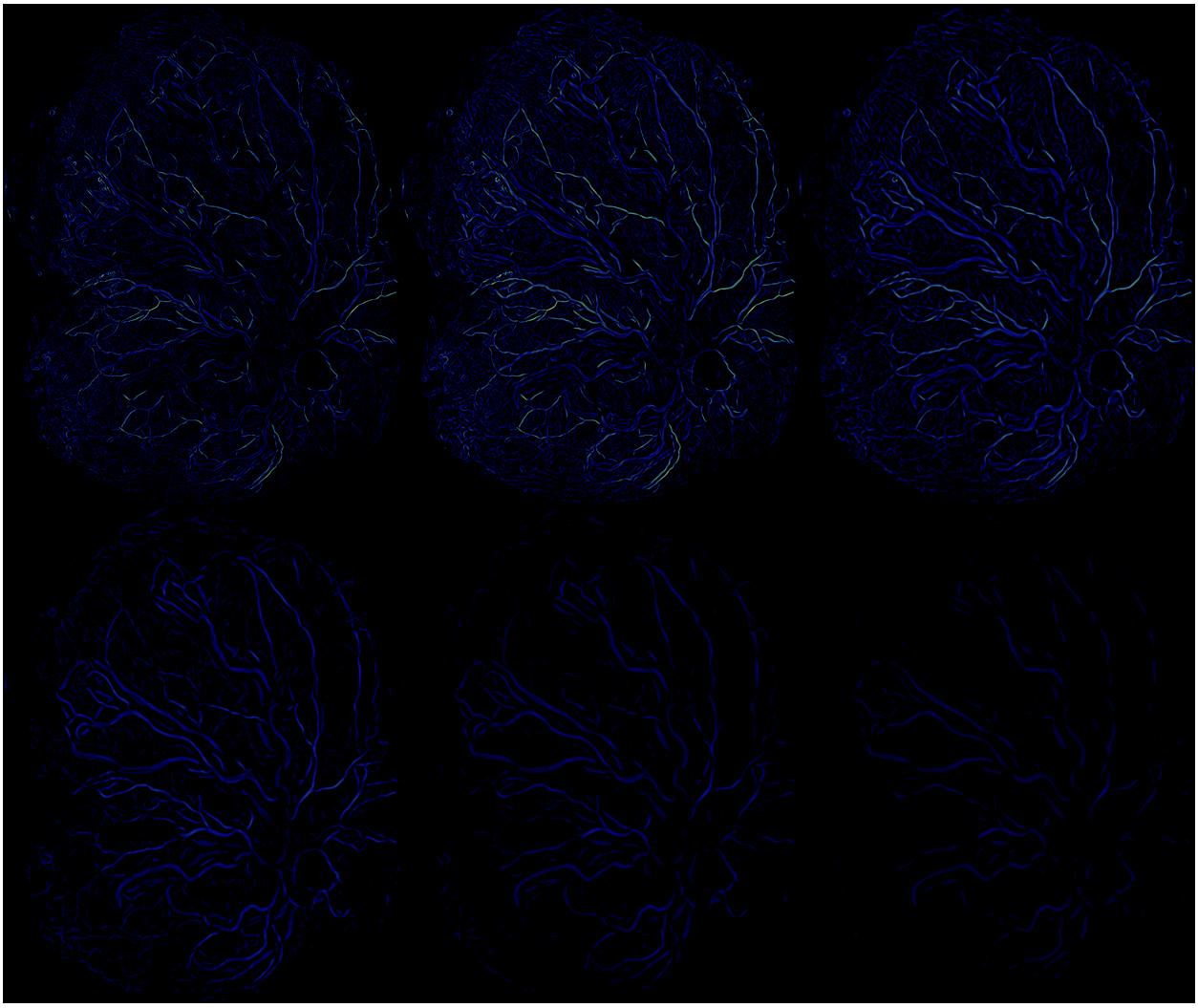
The smallest one should be an effective size where details are expected to be found, and the largest should be an effective size as well. In fact, following **[TODO: CITE REF]**, it is reasonable to select these logarithmically; that is, for some selected inputs $m < M$ we have

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

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" see section **[TODO: add ref]**.

Convolve this via fft transform to get L_{σ_i}



Applying Vesselness Measure

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

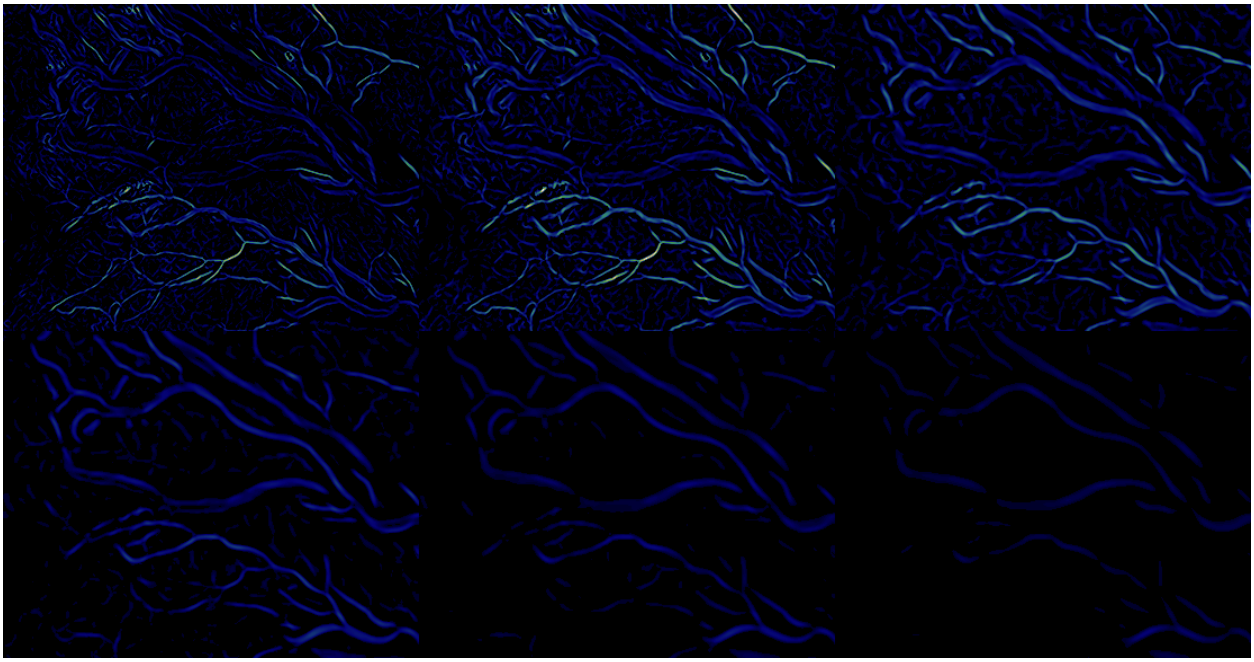
[TODO: Show at this point or move elsewhere how features of size σ are isolated in this step at the appropriate scale.]

Scale-space post-processing

Multiscale Merging

Cleanup/Postprocessing

Measurements



NOTE DUMP

[this is just a place to put commentary to sort/ rewrite later]

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.

Code Listings

[TODO: this goes in the appendix]

CHAPTER 5

RESULTS AND ANALYSIS

Data Set

Specifics of your data set. Barium placentas, other datasets.

NOTE: FIND YOUR TRACING PROTOCOL.

NOTE: Be specific about each so it's easy to gauge for others whether or not these methods would apply to their own research

Preprocessing

Results

Show stuff who the fuck knows

Answer Research Questions

CHAPTER 6

CONCLUSION

What it did well. What it didn't. People who want to expand–where should they start?

APPENDICES

APPENDIX A

APPENDIX TITLE

Put code here (and on github)

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] 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.

Starting point.

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

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

[4] 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.