BACHELOR

Improving on vascular tracking using sub-Riemannian geodesics

Smets, Bart M.N.

Award date:
2017

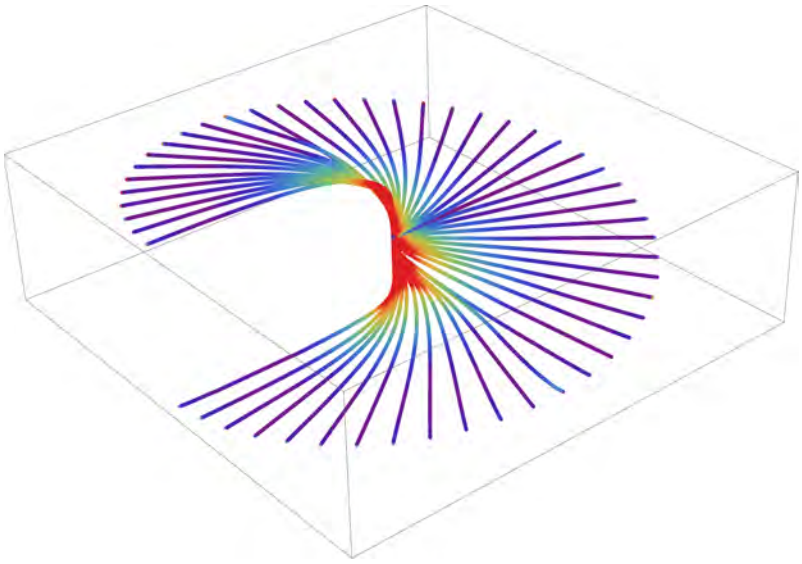
Technische Universiteit
Eindhoven
University of Technology

Faculty of Mathematics and Computer Science

Improving on Vascular Tracking using Sub-Riemannian Geodesics

Bachelor's Thesis

B.M.N. Smets



Supervisors:
dr.ir. R. Duits
dr.ir. E. Bekkers

July 26, 2017

Contents

Preface

In this report we aim to detail the methods we developed to improve vascular tracking methods using sub-Riemannian geodesics. In chapter 1 we introduce the theoretical background for these tracking methods. In particular we detail P-curve geodesics for uniform cost before introducing *Fast Marching* as a general numerical approach to finding geodesics for non-uniform cost with different metrics.

In chapter 2 we detail how we go about building a cost-function from a source image using orientation score transforms and what options we have in performing tracking on those cost-functions.

In chapter 3 we develop an analysis-tool for geodesics that we then use to find keypoints such as crossings, bifurcations and tracking-deviations as well as improve connecting vessel tips with their corresponding seeds.

We use chapter 4 for brief concluding remarks about the effectiveness of the methods, the improvements to the implementation we made along the way and make some suggestions for future work.

Chapter 1

Geodesics in SE(2)

In this chapter we will detail how we can see curves in the plane (and thus vessel traces in an image) as more complex geometry inside $SE(2)$. We can analyze such curves with the rich tools of differential geometry. In particular we are interested in different varieties of geodesics as we can think of blood-vessels as the shortest path for the blood to follow given a certain metric and vascular geometry.

We start by detailing the nature of $SE(2)$ and how we can *lift* curves in the plane to curves in $SE(2)$.

1.1 Lifting Curves in the Plane

In the classic Euclidean view, curves in the plane are seen as a 1-dimensional continuous set of points. By adding a Cartesian coordinate system we identify points with pairs (x, y), after which we can parametrize the curve as a function of $\mathbb{R} \to \mathbb{R}^2$ with some parameter u. When we are interested in the direction of the curve at a certain point we can obtain the tangent to the curve via the derivatives of the coordinate functions (at least on the points of the curve where those derivatives exist).

If we wanted to give the direction of the curve a more prominent place in our calculations we could dispense with derivatives and include it directly in our parameterization by introducing a triple of coordinates $(x(u), y(u), \theta(u))$, where we retain the conventional Cartesian coordinates x and y and use θ to denote the direction of the curve at that point with respect to the same Cartesian coordinate system.

Although not strictly necessary we will be using the common mathematical convention for dealing with direction θ:

- θ is measured in radians
- $\theta = 0$ is in the direction of the positive x-axis
- increasing θ changes the direction anti-clockwise

Formally we define $\theta(u) = \arg(\dot{x}(u) + i\ \dot{y}(u))$. A consequence of this is that directions are equal modulus 2π i.e.:

$$\theta_1 = \theta_2 \iff (\theta_1 - \theta_2) \mod 2\pi = 0$$

A parameterization of a curve now becomes a function of $\mathbb{R} \to \mathbb{R}^2 \rtimes S^1$, the latter which can be interpreted as the set of all translations and rotations in a plane, also called the Special Euclidian group of 2 dimensions, or SE(2) for short.

In figure 1.1 we illustrate how two curves are *lifted* from $\mathbb{R}^2$ to $SE(2)$. Note that while the curves intersect in $\mathbb{R}^2$ they do not do so in $SE(2)$.

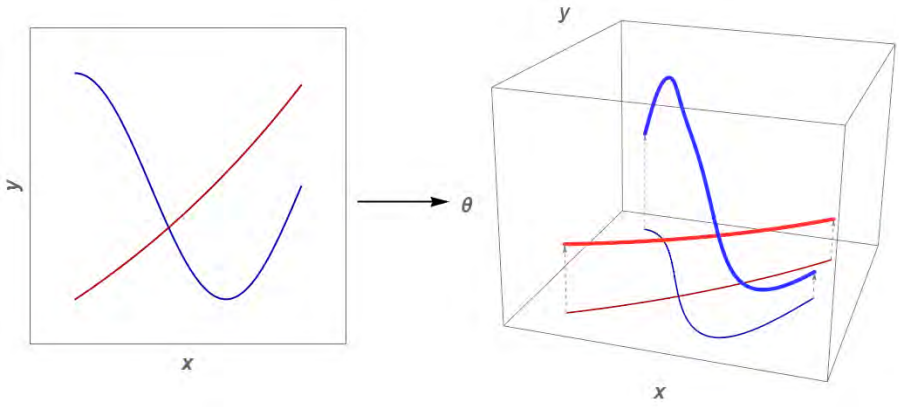


Figure 1.1: An example of *lifting* curves from $\mathbb{R}^2$ to $SE(2)$

1.2 Spatial Interpretation of $SE(2)$

We have already decided to express direction in unit-less radians, but the spatial coordinates (x, y) will generally represent some physical unit of distance related to the object of study. Since we intend to study digital images we will assume the spatial unit is a pixel going forward without saying how such a pixel relates to the physical dimensions of the object in the image.

To imagine $SE(2)$ as a dimensionally coherent 3D volume we need to relate radians to the spatial dimensions in some fashion. Most straightforward would be the introduction of a proportionality constant with dimension $[Length]^{-1}$, which in our case would amount to *radians/pixel*, we will denote this constant as ξ. In this fashion we can make any spatial distance dimensionless by multiplication with ξ.

Our choice of ξ can in principle be any positive real number, the exact value we choose would depend on how much we want to weigh change of direction against spatial movement. We can think of ξ as determining how *tall* we make $SE(2)$, where smaller ξ make for a taller volume which we illustrate in figure 1.2, where we can see that as ξ becomes smaller there is a larger 'distance' to cover for a given change of direction.

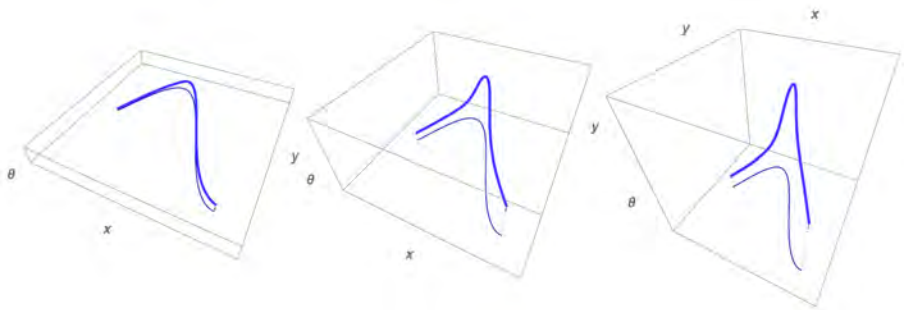


Figure 1.2: The effect of decreasing ξ on the spatial interpretation of $SE(2)$.

1.3 The P_{curve} Problem

Now that we have a geometric interpretation of $SE(2)$ we can proceed to think about geodesics. What we want to do is connect two elements of $SE(2)$ (i.e. a start- and end-point with associated directions) in an optimal way according to some measure which is generally stated as an Euler-Lagrange style functional. In this section we will look at the $\mathbf{P}_{curve}$ geodesic as detailed in [1].

1.3.1 Introduction

The $\mathbf{P}_{curve}$ problem is concerned with finding a sufficiently smooth[1] curve $\boldsymbol{\gamma} : [0, l] \to \mathbb{R}^2$ that given the start-point (x_s, y_s, θ_s) and end-point (x_e, y_e, θ_e) adheres to the following:

$$\begin{cases} \boldsymbol{\gamma} \in C^\infty([0,l], \mathbb{R}^2), \\ \boldsymbol{\gamma}(0) = (x_s, y_s), \quad \dot{\boldsymbol{\gamma}}(0) = (\cos\theta_s, \sin\theta_s), \\ \boldsymbol{\gamma}(l) = (x_e, y_e), \quad \dot{\boldsymbol{\gamma}}(l) = (\cos\theta_e, \sin\theta_e). \end{cases} \tag{1.1}$$

We have some freedom in choosing what parameterization to use, we will rely on the spatial arclength parameterization, which we denote by using parameter s. The spatial arclength parameterization has the following defining characteristic:

$$\dot{x}(s)^2 + \dot{y}(s)^2 = 1. \tag{1.2}$$

The curve we are now looking for is a minimizer of the following energy integral (where the total spatial arc-length l is free):

$$\boldsymbol{\gamma} = \underset{\boldsymbol{\gamma} \text{ per } (1.1)}{\operatorname{argmin}} \int_0^l \sqrt{\xi^2 + \kappa(s)^2} ds. \tag{1.3}$$

Where $\kappa(s)$ denotes the curvature as a function of the spatial arclength parameter s:

$$\boldsymbol{\gamma}(s) = \begin{pmatrix} x(s) \\ y(s) \end{pmatrix} \Rightarrow \kappa(s) = \frac{\dot{x}(s)\ddot{y}(s) - \ddot{x}(s)\dot{y}(s)}{(\dot{x}(s)^2 + \dot{y}(s)^2)^{\frac{3}{2}}}. \tag{1.4}$$

1.3.2 Scaling Invariance

Important for an imaging application is scaling invariance: the solution can not depend on the choice of resolution or unit of distance used. A solution to the $\boldsymbol{P}_{curve}$ problem exhibits this scaling invariance provided we scale ξ appropriately. We can in fact use clever rescaling to reduce the problem to one where $\xi = 1$, thus simplifying the problem somewhat. We state this formally in the following theorem, where we denote the scaling operator with factor λ as S_λ:

Theorem 1. *Given a curve $\boldsymbol{\gamma}(s)$ parametrized by it's arclength, that:*

- *Meets the boundary counditions* (1.1) *with* $(x_s, y_s) \neq (x_e, y_e)$
- *Is the global (unique) minimizer of the energy integral* (1.3) *for any value of* l

Then $S_\xi \boldsymbol{\gamma}$ is the global (unique) minimizer of the energy integral $\int_0^{l_{new}} \sqrt{1 + \kappa_{new}^2} ds_{new}$ with boundary conditions:

[1] Though we might cast a wider net initially we restrict ourselves to C^∞ here to head off any potential technical problems such as abnormal extreme, see [2] for details.

- $S_\xi \boldsymbol{\gamma}(0) = (\xi x_s, \xi y_s), \qquad S_\xi \boldsymbol{\gamma}(\xi l) = (\xi x_e, \xi y_e),$
- $\frac{d}{ds_{new}} S_\xi \boldsymbol{\gamma}(0) = \frac{d}{ds}\gamma(0), \qquad \frac{d}{ds_{new}} S_\xi \boldsymbol{\gamma}(\xi l) = \frac{d}{ds}\gamma(l).$

Proof.
Scaling with factor ξ changes arc-length and the coordinate functions in a straightforward manner:

$$\begin{aligned} l_{new} &= \xi l, \qquad s_{new} = \xi s, \\ x_{new}(s_{new}) &= \xi x(s), \qquad y_{new}(s_{new}) = \xi y(s). \end{aligned}$$

The curvature is similarly modified:

$$\begin{aligned} \kappa_{new}(s_{new}) &= \frac{\dot{x}_{new}(s_{new})\ddot{y}_{new}(s_{new}) - \ddot{x}_{new}(s_{new})\dot{y}_{new}(s_{new})}{\left(\dot{x}_{new}(s_{new})^2 + \dot{y}_{new}(s_{new})^2\right)^{3/2}} \\ &= \frac{\xi\dot{x}(s)\xi\ddot{y}(s) - \xi\ddot{x}(s)\xi\dot{y}(s)}{\left(\xi^2\dot{x}(s)^2 + \xi^2\dot{y}(s)^2\right)^{3/2}} \\ &= \frac{\xi^2}{\xi^3} \cdot \frac{\dot{x}(s)\ddot{y}(s) - \ddot{x}(s)\dot{y}(s)}{\left(\dot{x}(s)^2 + \dot{y}(s)^2\right)^{3/2}} \\ &= \frac{1}{\xi} \cdot \kappa(s). \end{aligned}$$

We now look at the energy integral:

$$\begin{aligned} E_\xi(\boldsymbol{\gamma}) &= \int_0^l \sqrt{\xi^2 + \kappa(s)^2} \cdot ds \\ &= \int_0^l \sqrt{\xi^2 + \xi^2\kappa_{new}(s_{new})^2} \cdot ds \\ &= \int_0^{\xi l} \xi\sqrt{1 + \kappa_{new}(s_{new})^2} \cdot \frac{1}{\xi} ds_{new} \\ &= \int_0^{l_{new}} \sqrt{1 + \kappa_{new}(s_{new})^2} \cdot ds_{new} \\ &= E_1(S_\xi\boldsymbol{\gamma}). \end{aligned}$$

We now argue that since scaling with factor $\xi > 0$ is a bijection that if $\boldsymbol{\gamma}$ is the global (unique) optimal solution to the initial problem then $S_\xi\boldsymbol{\gamma}$ is the same for the rescaled problem where $\xi = 1$. If there was another solution $\boldsymbol{\phi}$ to the rescaled problem with $E_1(\boldsymbol{\phi}) \leq E_1(S_\xi\boldsymbol{\gamma})$ then $E_\xi(S_\xi^{-1}\phi) \leq E_\xi(\boldsymbol{\gamma})$, a contradiction since we assumed $\boldsymbol{\gamma}$ was the unique minimizer[2].

■

This theorem proves remark 1.2 made by Duits *et al.* in [1] and allows us to restrict ourselves to studying these types of curves for $\xi = 1$.

1.3.3 Cusps

The spatial arc-length parameterization of a P-curve breaks down with the appearance of *cusps*, points where the curve stops and reverses on itself. This makes $(\dot{x}, \dot{y}) = (0, 0)$ at some point which is incompatible with (1.2).

[2]Since C^∞ is invariant under scaling we are not optimizing in different sets.

Assuming we have rescaled the problem so that $\xi = 1$ we can proceed to find at what spatial distance the curve reaches its (first) cusp, a distance we will call s_{max}. To aid us we first introduce the concept of normalized curvature z as follows:

$$z(s) = \frac{\kappa(s)}{\sqrt{1+\kappa(s)^2}}. \tag{1.5}$$

We now claim the following:

Theorem 2. *A solution to the $\boldsymbol{P}_{curve}$ problem (with $\xi = 1$) that has $z(0) = z_0$ and $\dot{z}(0) = \dot{z}_0$ has a (first) cusp at a distance of:*

$$s_{max} = \log\left(\frac{1+\sqrt{-z_0^2+\dot{z}_0^2+1}}{|z_0+\dot{z}_0|}\right). \tag{1.6}$$

Proof.
Note that $z \to \pm 1$ as $\kappa \to \pm\infty$ and so a cusp is reached when $|z(s)| = 1$.

Using variational techniques we can derive the Euler-Lagrange equation for our problem (1.3) by perturbing the curve along its normal and finding the stationary curve(s) for the following functional:

$$E_1(\boldsymbol{\gamma}(s) + \varepsilon\delta(s)\boldsymbol{n}(s)). \tag{1.7}$$

Where ε is an arbitrarily small real number, $\delta(s) : \mathbb{R} \to \mathbb{R}$ is an arbitrary but well-behaved perturbation function and $\boldsymbol{n}(s)$ signifies the normal to the curve. We refer to [1, Appendix A] for a derivation and limit ourselves to stating the relevant Euler-Lagrange equation in terms of z:

$$\frac{d^2z}{ds^2}(s) = z(s). \tag{1.8}$$

Which yields the general solution:

$$z(s) = z_0\cosh(s) + \dot{z}_0\sinh(s). \tag{1.9}$$

Clearly $z(0) = z_0$ and $\dot{z}(0) = \dot{z}_0$, so these constants derive from the boundary conditions. We can now identify the first cusp of the curve with the first positive solution of $|z(s)| = 1$:

$$\begin{aligned} &\left|\frac{z_0}{2}(e^s+e^{-s}) + \frac{\dot{z}_0}{2}(e^s-e^{-s}))\right| = 1 \qquad (1.10)\\ \iff & z_0(e^s+e^{-s}) + \dot{z}_0(e^s-e^{-s}) = \pm 2\\ \iff & (z_0+\dot{z}_0)e^{2s} \pm 2e^s + (z_0-\dot{z}_0) = 0\\ \iff & e^s = \frac{\pm 1 \pm \sqrt{-z_0^2+\dot{z}_0^2+1}}{z_0+\dot{z}_0}. \end{aligned}$$

We see that if $z_0 = -\dot{z}_0$ then the distance to the first cusp is $+\infty$, i.e. there is no first cusp. Secondly the sign of $z_0 + \dot{z}_0$ only serves to flip the $\pm$ symbols so we can take the absolute value without eliminating solutions:

$$\iff e^s = \frac{\pm 1 \pm \sqrt{-z_0^2+\dot{z}_0^2+1}}{|z_0+\dot{z}_0|}.$$

In which we have 4 options for choosing signs, since we are looking for positive solutions for s (so $e^s \geq 1$) we are limited to the following:

$$\iff e^s = \frac{\pm 1 + \sqrt{-z_0^2+\dot{z}_0^2+1}}{|z_0+\dot{z}_0|} \qquad \text{if } \pm 1 + \sqrt{-z_0^2+\dot{z}_0^2+1} \geq |z_0+\dot{z}_0| \geq 0.$$

The function $\pm 1 + \sqrt{-z_0^2 + \dot{z}_0^2 + 1}$ has a global minimum of ± 1 depending on the choice of sign, clearly then $(+)$ is the only correct choice if we want to meet the above conditional. We are left with a single positive solution:

$$\iff s = \log\left(\frac{1 + \sqrt{-z_0^2 + \dot{z}_0^2 + 1}}{|z_0 + \dot{z}_0|}\right) \equiv s_{max}. \tag{1.11}$$

■

This proves equation (78) given by Duits *et al.* in [1, Appendix A] and indicates up to what distance the spatial arc-length parameterization can be relied on.

1.4 Numerical Approach

In practice we will want to obtain geodesics numerically, indeed any hopes of obtaining analytical results will rapidly go out the window after we introduce image-driven geometry to the mix. In this section we will describe how we obtain numerical geodesics for more general geometry then the $\boldsymbol{P}_{curve}$ problem we discussed above. For a more thorough treatment of this topic we refer to [3] and [4].

1.4.1 General Finsler Geodesics in $\mathbb{R}^2 \rtimes S^1$ and $\mathbb{R}^2 \rtimes P^1$

We already detailed the $\mathbb{R}^2 \rtimes S^1$ manifold, the $R^2 \rtimes P^1$ manifold is essentially a subset of $SE(2)$ that equates antipodal points:

$$\mathbb{R}^2 \rtimes P^1 = \mathbb{R}^2 \rtimes S^1/2 \quad \text{with} \quad \boldsymbol{n}_1 \sim \boldsymbol{n}_2 \iff \boldsymbol{n}_1 = \pm \boldsymbol{n}_2 \tag{1.12}$$

We also refer to this manifold as $PT(\mathbb{R}^2)$. In this section we will write M when we mean either of these two manifolds.

Finsler geodesics are curves that minimize an energy integral of the following form:

$$d_{\mathcal{F}}(\boldsymbol{x}_s, \boldsymbol{x}_e) = \inf_{\substack{\boldsymbol{\gamma} \in Lip([0,1], M) \\ \boldsymbol{\gamma}(0) = \boldsymbol{x}_s \\ \boldsymbol{\gamma}(1) = \boldsymbol{x}_e}} \int_0^1 \mathcal{F}(\boldsymbol{\gamma}(t), \dot{\boldsymbol{\gamma}}(t)) \cdot dt \tag{1.13}$$

The pseudo-metric $\mathcal{F}$, which we refer to as the Finsler function, has the following properties:

(1) $\mathcal{F}(\boldsymbol{x}, \boldsymbol{v}) \geq 0$ (non-negative)

(2) $\mathcal{F}(\boldsymbol{x}, \boldsymbol{v}) = 0 \iff \boldsymbol{v} = \boldsymbol{0}$ (definite)

(3) $\forall \lambda \geq 0 : \mathcal{F}(\boldsymbol{x}, \lambda \boldsymbol{v}) = \lambda F(\boldsymbol{x}, \boldsymbol{v})$ (positive homogeneous)

(4) $\mathcal{F}(\boldsymbol{x}, \boldsymbol{v} + \boldsymbol{w}) \leq \mathcal{F}(\boldsymbol{x}, \boldsymbol{v}) + \mathcal{F}(\boldsymbol{x}, \boldsymbol{w})$ (subadditive)

(5) $\mathcal{F}(\boldsymbol{x}, \boldsymbol{v}) \geq \|\boldsymbol{v}\|$ (well-posed with respect to the Euclidian norm $\|\cdot\|$)

Note that we do not require symmetry: in general $\mathcal{F}(\boldsymbol{x}, \boldsymbol{v}) \neq \mathcal{F}(\boldsymbol{x}, -\boldsymbol{v})$.

For our application we will be using Finsler functions of a particular form, namely:

$$\mathcal{F}(\boldsymbol{x}, \boldsymbol{v}) = \mathcal{C}(\boldsymbol{x}) \cdot \mathcal{M}_{\xi}(\boldsymbol{x}, \boldsymbol{v}). \tag{1.14}$$

The function $\mathcal{C} : M \to [1, c_{max}]$, which we refer to as the cost-function, is what we derive from an image and associates a cost with each particular location (x, y, θ) in $SE(2)$ (or $\mathbb{R}^2 \rtimes P^1$ as is the case). Thinking back to an image, it gives a cost of traversing a particular point of the image in a particular direction. To meet the *positive definite* and *well-posedness* criteria of the Finsler function we require that the cost-function is normalized to a range of $[1, c_{max}]$ for some $c_{max} \geq 1$.

The second part of the Finsler function is the pseudo-metric $\mathcal{M}_\xi : T(M) \to \mathbb{R}_0^+$ where:

$$T(M) = \{(\boldsymbol{x}, \boldsymbol{v}) \mid \boldsymbol{v} \in T_{\boldsymbol{x}}(M)\}. \tag{1.15}$$

It associates a cost not with spatial position but with the direction and speed of travel along the curve, not only spatially but also (and more importantly) in the θ dimension thus associating a cost with change of direction. Since the pseudo-metric needs to be dimensionally consistent it needs to depend on ξ to make the spatial components dimensionless, hence its inclusion in the notation.

In terms of the car analogy $\mathcal{C}$ can be thought of as defining the geography over which the car is expected to drive and $\mathcal{M}_\xi$ controls the driving characteristics of the car. We will introduce the method we used to construct $\mathcal{C}$ in the next chapter, in the remainder of this section we will restrict ourselves to the constant cost case $\mathcal{C} \equiv 1$ and explore different choices for $\mathcal{M}_\xi$.

The first and most simple choice of pseudo-metric is the Euclidian, which we will call isotropic 2D since it acts only on the spatial dimensions and does not prefer any direction, note that it is not strictly necessary to include ξ here for the purpose of computation, but we include it for dimensional consistency:

$$\mathcal{M}_\xi^{(Iso2D)}(\boldsymbol{\gamma}(\tau), \dot{\boldsymbol{\gamma}}(\tau)) = \xi\sqrt{|\dot{x}(\tau)|^2 + |\dot{y}(\tau)|^2} \tag{1.16}$$

A second approach involves taking into account the θ dimension:

$$\mathcal{M}_\xi^{(SE(2))}(\boldsymbol{\gamma}(\tau), \dot{\boldsymbol{\gamma}}(\tau)) = \begin{cases} \sqrt{\xi^2|\dot{x}(\tau)|^2 + \xi^2|\dot{y}(\tau)|^2 + \left|\dot{\theta}(\tau)\right|^2} \\ \qquad\qquad \text{if } \exists\lambda \in \mathbb{R} : (\dot{x}, \dot{y}) = \lambda(\cos\theta, \sin\theta) \\ +\infty \qquad \text{otherwise} \end{cases} \tag{1.17}$$

As opposed to the isotropic pseudo-metric this one is not Riemannian since we do not allow lateral movement by assigning an infinite cost. We also see that the inclusion of ξ has become essential not just for the dimensional consistency of the pseudo-metric but also for the final value of the integral.

A refinement we can make to the last pseudo-metric is not allowing backward motion as follows:

$$\mathcal{M}_\xi^{(SE(2)+)}(\boldsymbol{\gamma}(\tau), \dot{\boldsymbol{\gamma}}(\tau)) = \begin{cases} \mathcal{M}_\xi^{(SE(2))}(\boldsymbol{\gamma}(\tau), \dot{\boldsymbol{\gamma}}(\tau)) & \text{if } \exists\lambda > 0 : (\dot{x}, \dot{y}) = \lambda(\cos\theta, \sin\theta) \\ +\infty & \text{otherwise} \end{cases} \tag{1.18}$$

We refer to this variant as forward- or positive-control and it eliminates the case of the regular $SE(2)$ metric where the curve is allowed to go in reverse.

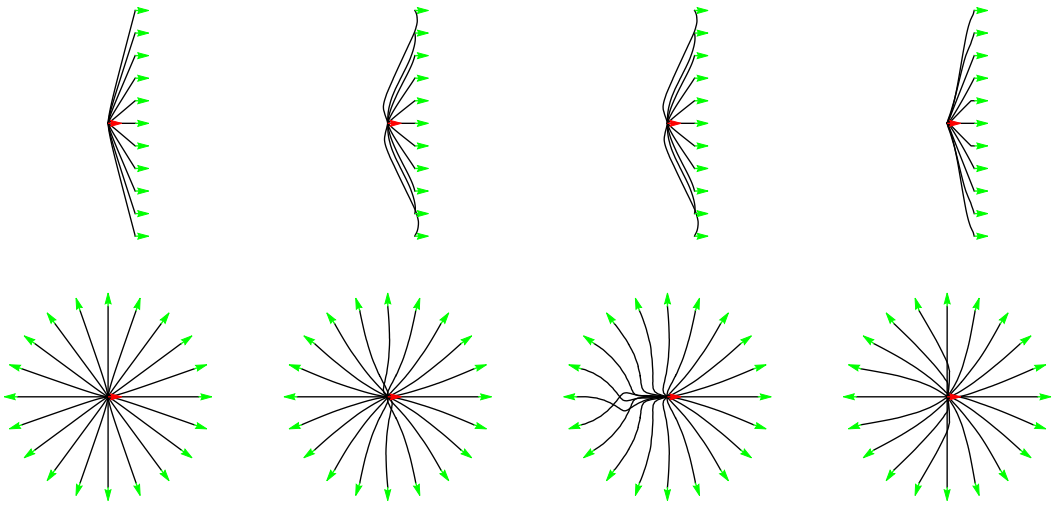
A final variant we will mention is the projective line bundle $PT(\mathbb{R}^2)$ model, in it we use the same metric as for the $SE(2)$ case but now equate antipodal points (i.e. we only look at direction and not orientation essentially discarding half of $SE(2)$). This has the effect of allowing curves to stop and reverse at no cost. For more details on this metric we refer to [5].

1.4.2 Fast Marching

The general numerical approach to this type of problem is called *Fast Marching* [6] which can be thought of as a volumetric version of Dijkstra's Algorithm for graphs. We won't detail the workings

but refer to [3] for further information. The implementation of the fast marching algorithms we used in this paper is called *HamiltonFastMarching* by Mirebeau *et al.* [7] and allows us to compute geodesics numerically after which we use *Mathematica* for visualization.

Figure 1.3 shows such visualizations for the four different metrics we introduced in the previous section applied to two test cases with a constant $\mathcal{C} = 1$.



(a) Isotropic (b) SR on PTR2 (c) SR on SE(2) (d) SR on SE(2) positive

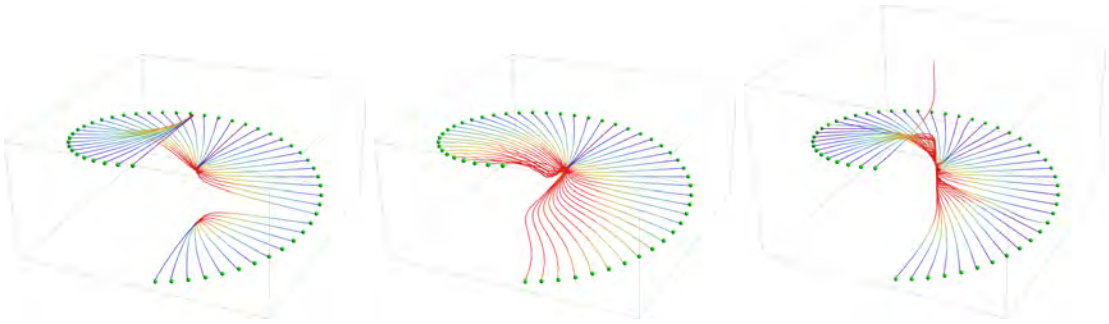
Figure 1.3: Numerical geodesics on a $200 \times 200 \times 32$ grid with $\xi = 0.02$ for different metrics.

Also interesting is how these geodesics can be visualized as lifted curves in $SE(2)$, we show the three circular test-cases from above as lifted curves in figure 1.4.

The colors are a measure of spatial curvature vs spatial velocity, with red indicating a higher ratio. We notice how the projective metric is just as happy to reverse out of its starting position towards its destination if that saves on distance travelled, hence why half the geodesics don't seem to start from the correct point in $SE(2)$ but are π of in the θ dimension.

In the regular $SE(2)$ metric the geodesics go through a fair bit of backward contortion to reach destinations behind their starting points. The positive-control variant seems to produce the most logical curves from a *segway*-like point of view: they turn towards their target and then start moving towards it spatially. Note the *broken piano string* in the last figure which is likely caused by a floating point modulus operation gone awry[3] which would account for the starting point being 2π off, otherwise it is correct.

[3]But we have not investigated further



(a) SR on PTR2 (b) SR on SE(2) (c) SR on SE(2) positive

Figure 1.4: Lifted geodesics on a $200 \times 200 \times 32$ grid with $\xi = 0.02$ for different sub-Riemannian metrics.

Chapter 2

Vessel Tracking

In this chapter we detail how we can construct $\mathcal{C}$, the cost function we introduced in the first chapter. Starting with orientation scores we will construct a vesselness-measure that is suitable for tracking vessels in $SE(2)$ with fast marching.

While the techniques we will detail can principally be used on color images by applying them to the different color channels, we will be restricting ourselves to gray-scale images.

The images we will use to demonstrate and validate these techniques are of the retina, consisting of 52 different so-called *patches* that show different isolated parts.

2.1 Orientation Scores

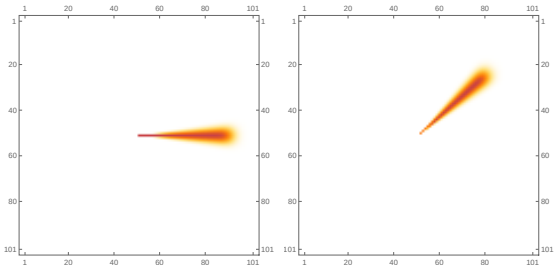
Orientation score transforms do for orientation and direction what local Fourier transforms (or: Gabor transforms) do for frequency: they decompose an image according to its response to a particular orientation. In this fashion an orientation score transform is a natural map from $\mathbb{R}^2 \rtimes S^1$ to $\mathbb{R}^+$ and thus complementary to the techniques we saw in the previous chapter after appropriate normalization. For more details on Orientation Scores see [8] and [9].

Decomposition by orientation is accomplished by a wavelet-like transform where we generate a stack of wavelets, one for each orientation of interest. Each wavelet then serves as a convolution kernel that is applied to the image. The wavelets are crafted to be anisotropically sensitive and so filter out features not aligned to its orientation.

While we have some choice in wavelets we will restrict ourselves to so-called *cake wavelets* in this paper. For a more thorough look into these wavelets and how they are constructed we refer to [10, Chapter 2.1].

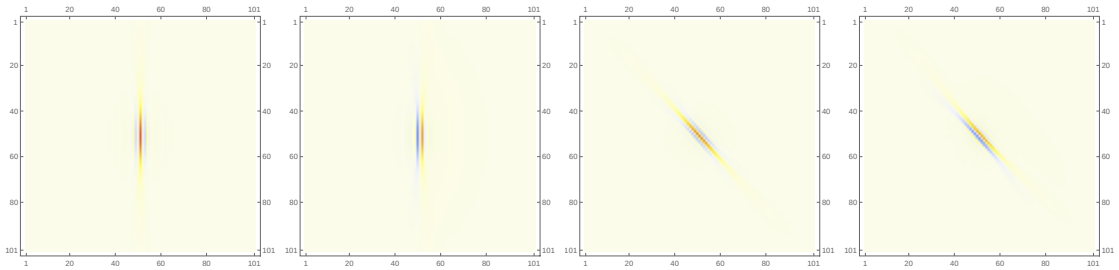
Cake wavelets borrow their name from looking like cake-pieces in the Fourier-domain which is illustrated in figure 2.1 for two directions of a 60 direction partitioning.

Filtering a particular slice of the Fourier domain in this manner will give the wavelets their anisotropic properties when they are transformed into the spatial domain as shown in figure 2.2. Since the wavelets were real-valued in the Fourier domain they are complex-valued spatially, which are plotted separately, shades of blue indicate positive values, shades of red indicate negative values.



(a) $\theta = 0$ (b) $\theta = \pi/4$

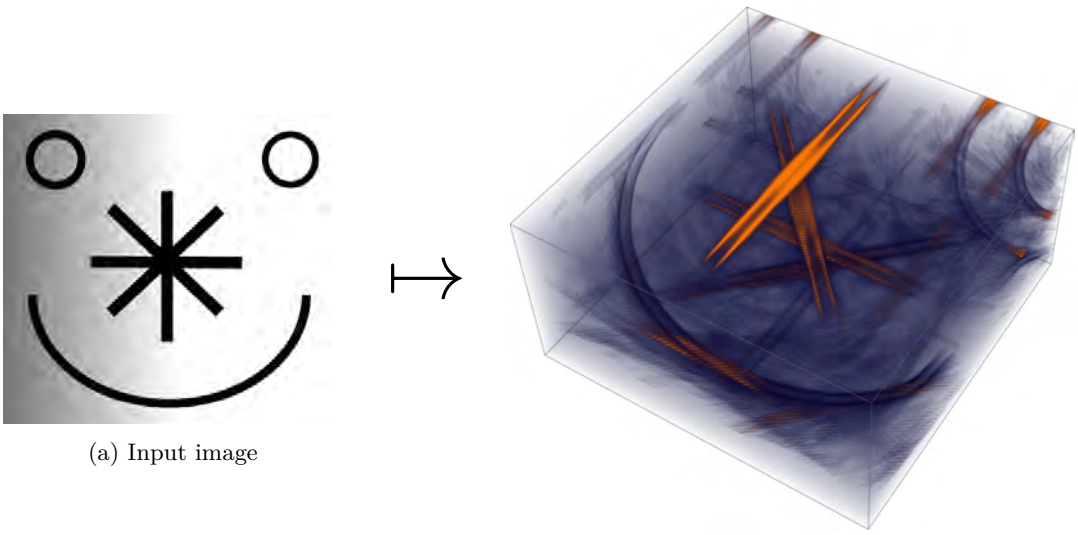
Figure 2.1: Cake wavelets in the Fourier domain ($101 \times 101 \times 60$) for two orientations.



(a) $\theta = 0$ Re. part (b) $\theta = 0$ Im. part (c) $\theta = \pi/4$ Re. part (d) $\theta = \pi/4$ Im. part

Figure 2.2: Cake Wavelets in the spatial domain ($101 \times 101 \times 60$) for two orientations.

The spatial wavelets are then used as convolution kernels applied to the image, separately for each orientation of the wavelets, real and imaginary parts. This yields a complex-valued volume referred to as the *Orientation Score* of the image. This transform is illustrated for a test-image in figure 2.3, notice how the central asterisk is pulled apart by the transform.



(a) Input image

(b) Orientation Score Volume

Figure 2.3: Orientation Score ($400 \times 400 \times 200$) of a test image with cake wavelets.

2.2 Vesselness and the Cost-function

Now we need to apply these techniques to actual images to construct the cost-function $\mathcal{C}$ that we can use to do tracking. We work the original gray-scale through the following steps to arrive at the cost-function:

1. Down-sample
2. Curvature-flow filter
3. Orientation Score
4. Vesselness-measure
5. Cost-function

While not necessary we generally down-sample input images to a reasonable size of around 250×250 to reduce computation times and memory usage. We then apply a Curvature-flow filter[1] to the bitmap image, which is an anisotropic diffusion filter that smooths the image while preserving edges, we do this to remove the noise from the image without blurring edges before applying the Orientation Score transform.

The vesselness measure is constructed as a weighted sum (by parameters σ) of smoothed derivatives at different scales of the Orientation Score U_f (per Zhang *et al.* [11]).

$$\mathcal{V}(x,y,\theta) = \sum_{i=1}^{n} \left(\frac{\sigma_{s_i}}{\sigma_o} \right)^{\gamma} \cdot \mathcal{G}^2_{2,(\sigma_{s_i},\sigma_o)} U_f(x,y,\theta). \tag{2.1}$$

These derivatives are taken via convolution with different sized Gaussian derivative kernels which we denote with $\mathcal{G}^2_{2,(\sigma_{s_i},\sigma_o)}$ by which which we mean the second Gaussian derivative (superscripted 2) orthogonal to the orientation[2] (subscripted 2) with spatial scale factors σ_{s_i} and an orientation scale factor σ_o.

Convolving with derivatives works here because the Gaussian kernels are anisotropic and so simplify the normal group action in $\mathbb{R}^2 \rtimes S^2$ to the same action as in $\mathbb{R}^2 \times S^1$:

$$(\boldsymbol{x},\theta)(\boldsymbol{x'},\theta_1) = (R_\theta \boldsymbol{x'} + \boldsymbol{x}, \theta + \theta') = (\boldsymbol{x} + \boldsymbol{x'}, \theta + \theta') \text{ provided that } R_\theta \boldsymbol{x'} = \boldsymbol{x'} \tag{2.2}$$

This makes the order of smoothing and derivation interchangeable (where G is the Gaussian kernel functions):

$$\begin{aligned}
&(-\sin\theta \cdot \partial_x + \cos\theta \cdot \partial_y)^2 \left(G^{\mathbb{R}^2}_{\sigma_s}(\boldsymbol{x}) G^{S^1}_{\sigma_o}(\theta) \underset{\mathbb{R}^2 \rtimes S^1}{*} U_f(\boldsymbol{x},\theta) \right) \\
=& (-\sin\theta \cdot \partial_x + \cos\theta \cdot \partial_y)^2 \left(G^{\mathbb{R}^2}_{\sigma_s}(\boldsymbol{x}) G^{S^1}_{\sigma_o}(\theta) \underset{\mathbb{R}^2 \times S^1}{*} U_f(\boldsymbol{x},\theta) \right) \\
=& \left((-\sin\theta \cdot \partial_x + \cos\theta \cdot \partial_y)^2 G^{\mathbb{R}^2}_{\sigma_s}(\boldsymbol{x}) \cdot G^{S^1}_{\sigma_o}(\theta) \right) \underset{\mathbb{R}^2 \times S^1}{*} U_f(\boldsymbol{x},\theta) \\
=& \mathcal{G}^2_{2,(\sigma_s,\sigma_o)} U_f(\boldsymbol{x},\theta)
\end{aligned} \tag{2.3}$$

The choice of parameters σ is somewhat dependent on the type of images one wishes to process, in our case we applied the following: $n = 10$, $\sigma s_i = i$, $\sigma o = 0.4$ and $\gamma = 2$. So we generate derivatives

[1] The filter used is standard to Mathematica, see `https://reference.wolfram.com/language/ref/CurvatureFlowFilter.html` for details.

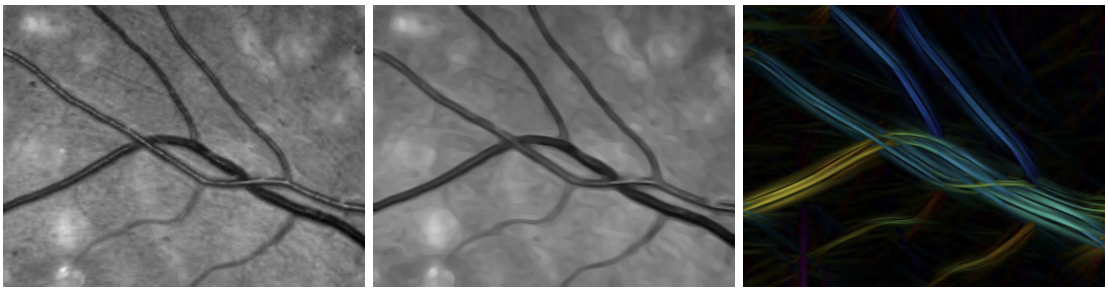
[2] As in: $-\sin\theta \cdot \partial_x + \cos\theta \cdot \partial_y$

at 10 scales where we run the Gaussian size over 1 to 10 pixels spatially and a constant 0.4 radians directionally. The weighting factor in front serves to provide a bias towards the larger scales.

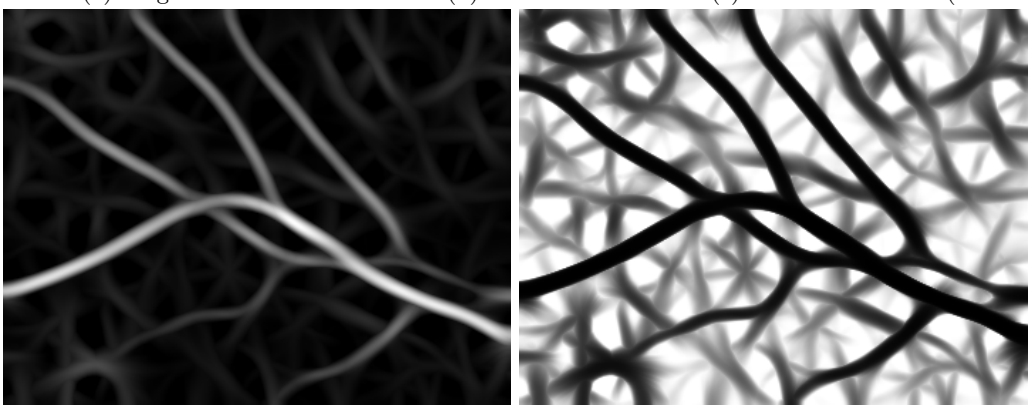
We perform a final transformation on the vesselness measure to aquire our cost function.

$$\mathcal{C}(x, y, \theta) = \frac{1}{1 + \lambda \cdot \mathcal{V}(x, y, \theta)^p}. \tag{2.4}$$

Where we use the parameters $\lambda = 200$ to control the strength of the cost function and $p = 2$ for the sharpness. Other choices of parameters could be desirable for different types of images. Figure 2.4 illustrates these steps taken on an example retinal image, note that the last three images are projected volumes for the sake of clarity on paper (the Orientation Score being additionally colored by orientation).



(a) Original (b) Pre-filter (c) Orientation Score (colorized)



(d) Vesselness (projected) (e) Cost Function (projected)

Figure 2.4: Image processing steps to construct $\mathcal{C}$

In some contexts the cost function is called the speed function, in which case it is simply the reciprocal of the cost function we described above (this is the case in the *HFM* package we use).

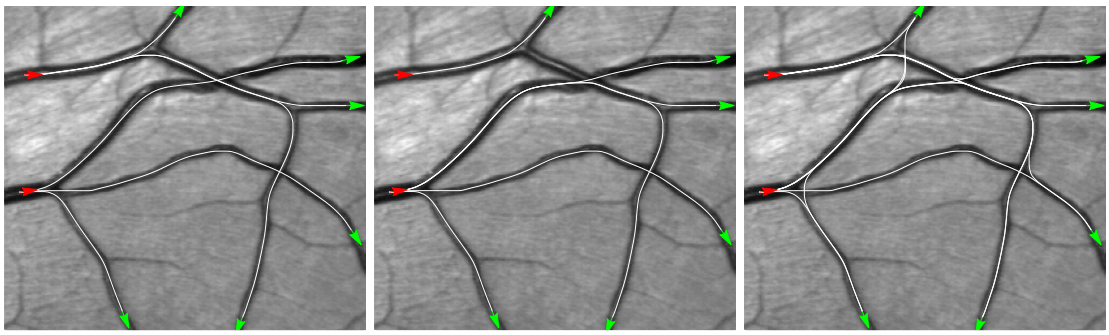
2.3 HFM Tracking

Now we have all the components needed to start tracking vessels in images. The *HFM* (Hamiltonian Fast Marching) software takes in the cost function and a set of starting point and ending points as boundary conditions. We will refer to these starting points as *seeds* and the ending points as *tips* from now on.

Given these seeds and tips the algorithm connects the latter with the former with the lowest possible cost according to the specified metric. This gives us a few options in how we do the tracking given multiple seeds and tips:

1. Grouped: do a tracking per seed and only include those tips that belong to that seed connectivity wise.
2. Combined: do a single tracking with all the seeds and tips together.
3. Tip-wise: do a tracking per tip and include all seeds.

Obviously the grouped tracking would be the most desirable if the meta-data needed to match seeds and tips is available, since this isn't the case in general we won't be using it. Instead we will see what we can learn from the other types of tracking to a) identify key-points b) infer the right matching between seeds and tips. The different tracking types are illustrated in figure 2.5, where seeds are indicated with red arrows and tips with green arrows.

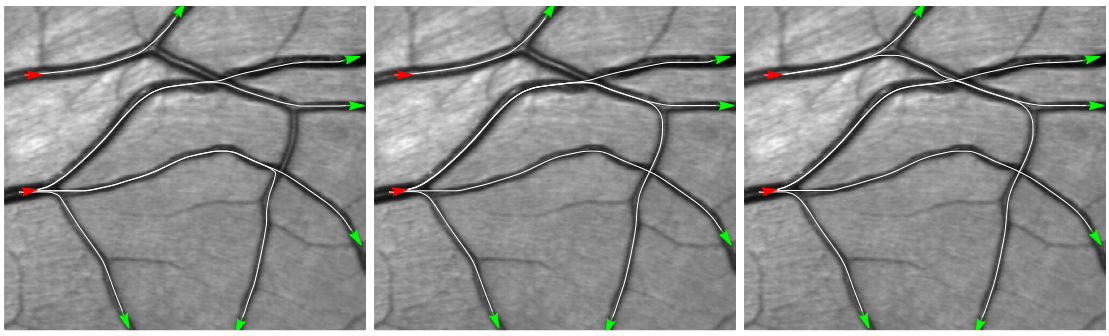


(a) Grouped (b) Combined (c) Tip-wise

Figure 2.5: Tracking with different seed/tip sets

Note that applying the full tracking type is already a form of seed/tip matching (in this case correctly matching 4 out of 6 tips), we will use it later to benchmark our developed method.

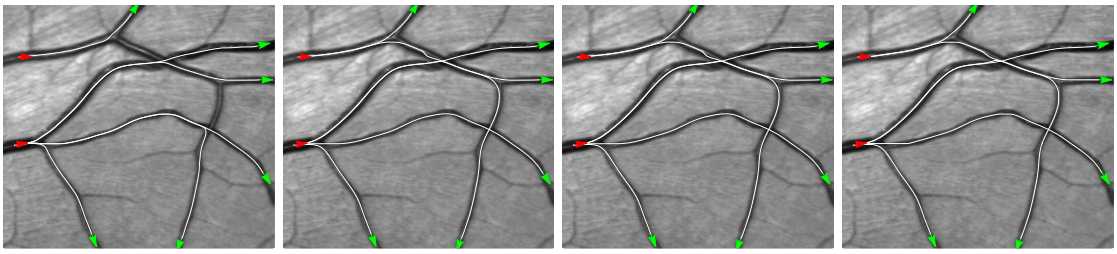
Modifying ξ also has a large impact on the quality of tracking, this is illustrated in figure 2.6. Note how lowering ξ to 0.01 in this case matches 6 out of 6 tips with their correct seeds. Lowering ξ too much causes excessive stiffness, making the traces not follow the vessels properly. For the images available to us $\xi = 0.02$ (excluding the down-sample factor) provided a reasonable trade-off and that is the value we will use through the rest of the paper unless otherwise mentioned.



(a) $\xi = 0.40$ (b) $\xi = 0.05$ (c) $\xi = 0.01$

Figure 2.6: Tracking with different values of ξ

Lastly, we have 4 available choices for metric, the effects of which are illustrated in figure 2.7. For appropriate parameterization the $PTR2$, $SE(2)$ and $SE(2)$ forward metric all perform well. In some more difficult images the $SE(2)$ forward metric generally provides some cleaner traces, which is why will be using it going forward as the default metric.



(a) Isotropic 2D (b) $PTR2$ (c) $SE(2)$ (d) $SE(2)$ Forward

Figure 2.7: Tracking with different metrics with $\xi = 0.01$

Chapter 3

Analyzing Traced Geodesics

In this chapter we will explore how we can analyze the obtained geodesics to detect keypoints as well as matching tips with their correct seeds. These two techniques, while not improving the tracings by themselves, are intended to serve as starting points for future techniques. We will base both these methods on the lift-measure we introduce first.

3.1 Lift-measure

We are interested in those pieces of the geodesics that are close to vertical in our spatial view of $SE(2)$, i.e. they have a high spatial curvature versus spatial velocity ratio. This ratio is simply a restatement of the spatial curvature κ scaled by ξ^{-1}, but in this form it is more easily applied to the discrete geodesics we have obtained. We wrap this ratio in an arctangent as follows to obtain our lift-measure:

$$\mathcal{L}_{\boldsymbol{\gamma},\xi}(\tau) = \arctan\left(\frac{|\dot{\theta}(\tau)|}{\xi\sqrt{\dot{x}(\tau)^2 + \dot{y}(\tau)^2}}\right) \tag{3.1}$$

The $\mathcal{L}_{\boldsymbol{\gamma},\xi}$ measure obviously depends on the curve $\boldsymbol{\gamma}(\tau) = (x(\tau), y(\tau), \theta(\tau))$, but also on ξ, which we require for dimensional consistency. We use the convention that if $\sqrt{\dot{x}(\tau) + \dot{y}(\tau)} = 0$ then the arctangent evaluates to $\frac{\pi}{2}$ (assuming $\dot{\theta}(\tau) \neq 0$). The introduction of the arctangent serves as a convenience so that we can express threshold values in the range $[0, \pi/2]$ instead of $[0, +\infty)$. In addition there is the interpretation of this measure as a vertical angle in $SE(2)$ that is intuitively appealing and also gives a geometric reason for the presence of ξ. Finally in the implementation we can use the two-argument version of the arctangent function eliminating the chance of a divide-by-zero and floating point overflows.

Figure 3.1 illustrates the lift-measure by coloring the geodesics according to the value of the measure, these figures incidentally depict tip-wise tracking.

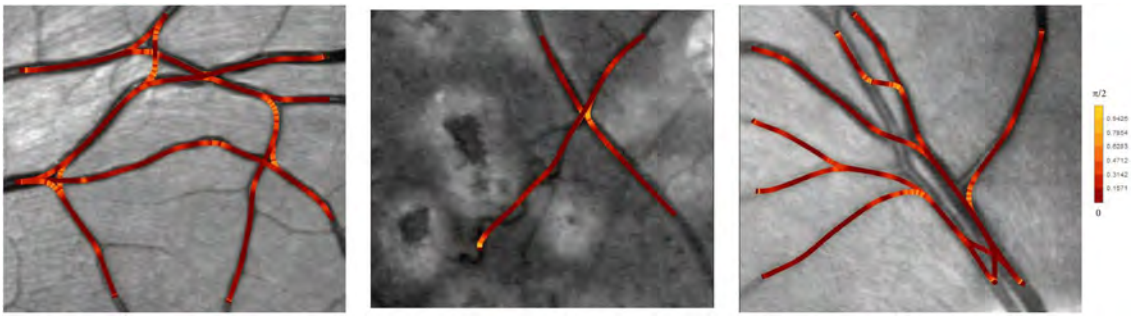


Figure 3.1: Vizualizing the lift-measure

To better deal with the obvious discretization noise visible in figure 3.1 we will apply some Gaussian smoothing on the geodesics before we calculate the measure. We use a window of 20 steps[1] with reflected padding to accomplish the smoothing without shortening the geodesics, the effect of this is illustrated in figure 3.2, where we see a much cleaner connection between color and curvature.

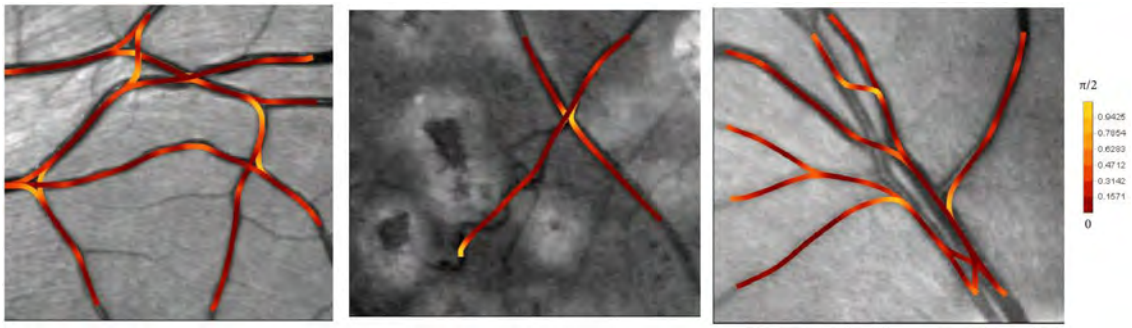


Figure 3.2: Vizualizing the lift-measure after Gaussian smoothing

We will also be using the following indicator version of the measure:

$$\mathcal{L}^{\alpha}_{\boldsymbol{\gamma},\xi}(\tau) \equiv \mathbb{1}\left\{\mathcal{L}_{\boldsymbol{\gamma},\xi}(\tau) \geq \alpha\right\} = \begin{cases} 1 & \text{if } \mathcal{L}_{\boldsymbol{\gamma},\xi}(\tau) \geq \alpha \\ 0 & \text{else} \end{cases} \tag{3.2}$$

3.2 Keypoint Detection

To detect keypoints we start by doing a tip-wise tracking. A tip-wise tracking has the downside of containing many errant geodesics that connect tips with wrong seeds (see figure 3.2), but has the benefit of definitely containing all the correctly matched geodesics as a subset. Moreover we are not really concerned with the geodesics as such, but with their local behaviour in the sort of points we want to detect.

After having applied the tracking we identify sections of the geodesics as having high lift-measure. We use the indicator version of the measure for this, the choice of threshold value α depends somewhat on what one wishes to accomplish, crossings and sharp bifurcations are readily detected for thresholds as high as $\pi/4$, shallow bifurcation generally only show up for lower values. For the images we had available $\alpha = \pi/7$ served as a good compromise, this is illustrated in figure 3.3.

[1] Geodesics typically consist of $300 - 800$ steps.

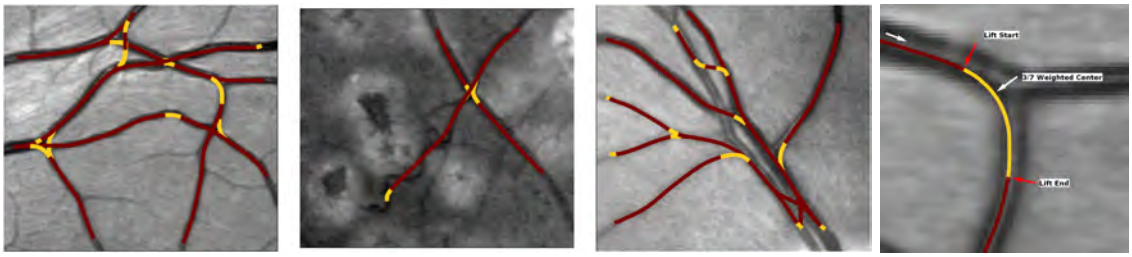


Figure 3.3: High measure sections for $\alpha = \pi/7$ on tip-wise tracking

Having identified sections of the geodesics as having high measure/curvature we have some freedom as to what point of that section we designate as the point-of-interest. For bifurcations the most accurate choice is choosing the starting-point, while crossings tend to happen in the middle of the lift. Designating the point 3/10ths of the way through the section as the point of interest served as a good compromise in all situations.

Having completed this operation we find ourselves with a set of points of interest derived from the tip-wise tracking, we can now use the geodesics of the *combined* tracking (tracking with all seeds and tips at once) to modify these points. The reasoning being that the combined tracking produces those geodesics that are most likely to correctly match seeds and tips according to the chosen metric. By moving our original set of points to to their nearest point on the combined geodesics improves their accuracy with respect to vascular features.

One apparent artifact of the tracking method is a consistent appearance of small sections of high measure at the seeds and/or tips, we will simply ignore these as not relevant.

We now have a set of points-of-interest located on the lowest cost geodesics connecting tips and seeds. In many cases a feature such as a bifurcation will cause two or more high measure sections to appear close to it, for that reason we apply a clustering algorithm to identify groups of points that are close to each other. A drawback is that two features that are close together might be identified as a single feature.

For identifying groups of lifts we have used the *Calinski-Harabasz* method[12] on the Euclidian distance. This method works by optimizing the ratio of the variance between the cluster centers and the variance between the members of a cluster.

Suppose we have n points with overall spatial center c, we then want to choose k clusters C_i, each with n_i elements and center c_i such that we maximize the following ratio:

$$\frac{\sum_i n_i \cdot |c - c_i|^2}{\sum_i \sum_{x \in C_i} |c_i - x|^2} \cdot \frac{n-k}{k-1} \tag{3.3}$$

We reduce the obtained clusters to points by taking the geometrical mean. This cluster center is likely again not on a geodesic of the combined set (or indeed any geodesic) so we apply the same remedy as previously and move the found cluster center to the nearest point of the combined geodesics.

We summarize the steps we took:

1. Perform tip-wise tracking
2. Smooth the found geodesics
3. Calculate lift-measure
4. Isolate high measure sections

5. Ignore high measure sections close to seeds/tips
6. Take 3/7-weighted center of sections
7. Move points to nearest *combined* geodesic
8. Identify clusters
9. Take cluster means
10. Move means to nearest *combined* geodesic

We illustrate the results of these steps in figure 3.4, the combined geodesics set is plotted, the orange dots are the raw detected points-of-interest and the red points are the final results of our detection procedure.

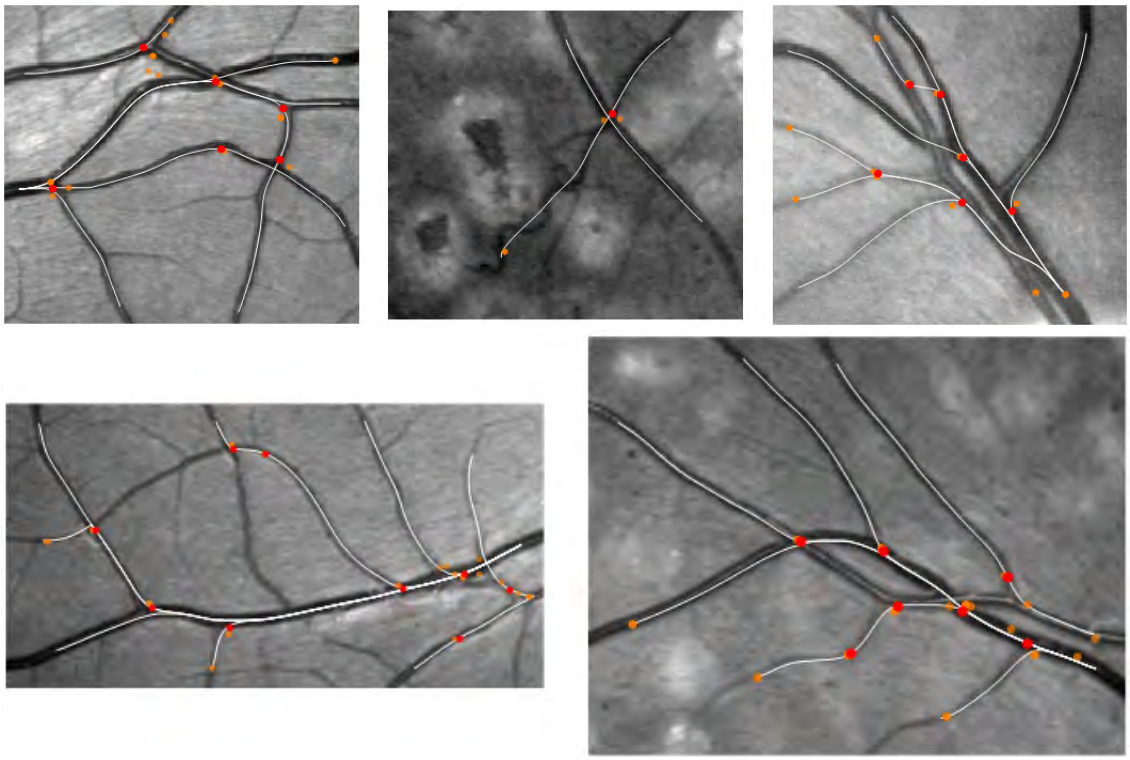


Figure 3.4: Results of keypoint detection (orange dots: raw, red dots: final)

3.3 Matching Seeds and Tips

In the previous section we used the lift-measure locally to identify keypoints, in this section we will use the lift-measure globally to choose which seed to match with a given tip.

In particular we want to calculate the total $SE(2)$-arc-length of the high measure sections on a geodesic given a certain threshold, which we calculate as follows with a given geodesic $\boldsymbol{\gamma} : [0,1] \to \mathbb{R}^2 \rtimes S^1$:

$$\mathcal{L}_{\xi}^{\alpha}\boldsymbol{\gamma} = \int_0^1 \mathcal{L}_{\boldsymbol{\gamma},\xi}^{\alpha}(\tau) \cdot \sqrt{\xi^2 \left(|\dot{x}(\tau)|^2 + |\dot{y}(\tau)|^2\right) + |\dot{\theta}(\tau)|^2} \cdot d\tau \tag{3.4}$$

Or if we have a geodesic that is parameterized by spatial arc-length:

$$\mathcal{L}_{\xi}^{\alpha}\boldsymbol{\gamma} = \int_{0}^{l} \mathcal{L}_{\boldsymbol{\gamma},\xi}^{\alpha}(s) \cdot \sqrt{\xi^2 + \kappa(s)^2} \cdot ds \tag{3.5}$$

We can use this global measure to choose between different geodesics reaching a single tip under the assumption that incorrect geodesics would have to jump a crossing somewhere and consequently significantly increase their measure. The procedure is as follows:

1. For a given tip track to each seed
2. Calculate local lift-measure over the obtained geodesics
3. Integrate to obtain global lift-measure
4. Select geodesic with lowest global measure

This is illustrated in figure 3.5 where we color the lowest measure geodesic blue and the highest red.

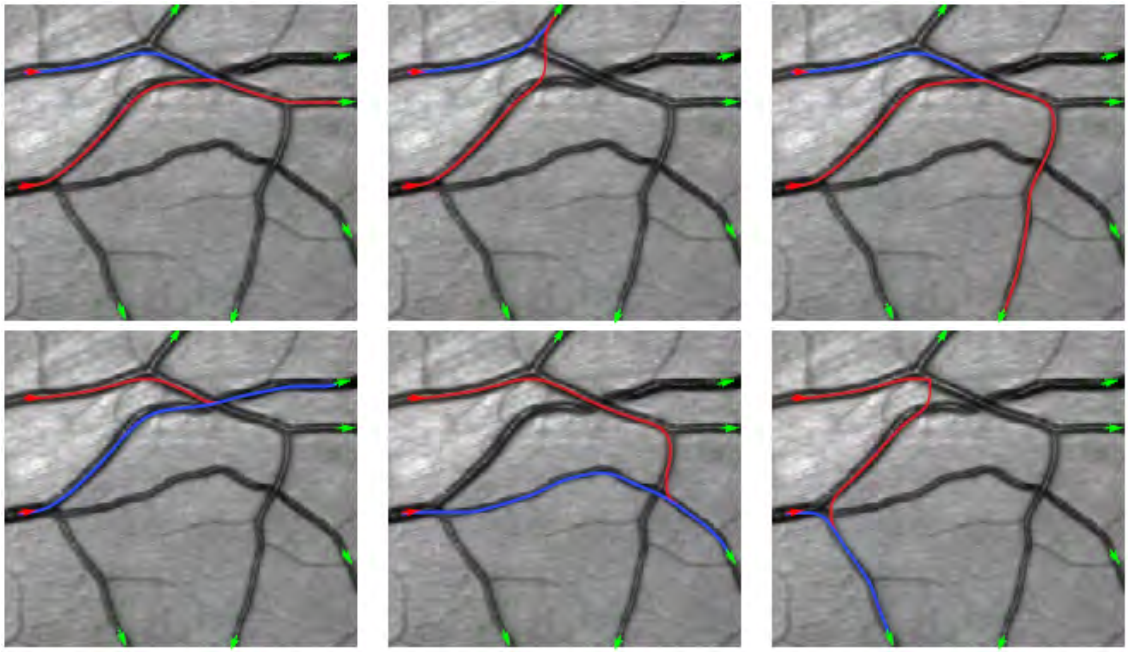


Figure 3.5: Matching different seeds to tips (blue=lowest measure)

To test the accuracy of the method we applied it to the 52 retina patches we had available, as a benchmark we compare it against how well combined tracking with the $SE(2)$ forward metric matches tips and seeds on its own. Results are summarized in table 3.1.

Number of images	52
Number of tips	301
$SE(2)$+ Fast Marching ($\xi = 0.02$)	193 correct (64%)
Lift-measure method ($\xi = 0.02$, $\alpha = \pi/4$)	208 correct (69%)

Table 3.1: Accuracy of $\mathcal{L}_{0.02}^{\pi/4}$ seed/tip matching

For more details and different approaches to the matching problem see [5].

Chapter 4

Concluding Remarks

On Improving Geodesic Tracking

The lift-measure $\mathcal{L}^{\alpha}_{\gamma,\xi}$ has proven a useful tool for identifying keypoints such as crossings, bifurcations and tracking-errors. Using the measure globally on the geodesics has also produced more reliable seed/tip matching.

Combined these techniques can serve as a basis for an improved tracking method based on sub-Riemannian geodesics.

On Performance

In the course of porting the Orientation Score Transform to the GPU (see appendix B) we noticed significant performance gains. Considering how computationally intensive many steps of the process are and how readily parallelizable they are we see GPUs as the way forward in making these techniques viable outside of academia.

Future Work

Some suggestions for future work:

- Extend the car-on-geography metaphor with an acceleration metric:

$$E_{\xi}(\boldsymbol{\gamma}) = \int_0^1 \mathcal{C}(\boldsymbol{\gamma}(\tau)) \cdot \mathcal{M}_{\xi}(\dot{\boldsymbol{\gamma}}(\tau)) \cdot \mathcal{N}_{\xi}(\ddot{\boldsymbol{\gamma}}(\tau)) \cdot d\tau$$

- Better discretization of cake-wavelets so that the response isn't higher for $\theta = \pi/2 \cdot \mathbb{Z}$ then for other orientations.
- Fine-tune the choices of parameters we made, in particular the lift threshold value α.
- Do a tracking with the seed/tip combinations we found with the help of $\mathcal{L}^{\alpha}_{\xi}$ and use that as the basis for a keypoint analysis.
- CUDA implementation:

  - Wishlist (see figure B.1, orange sections)
  - Investigate parallelization of Fast Marching algorithm
  - Performance benchmarking
- Clustering with sub-Riemannian metric approximations instead of Euclidian metric

Appendix A

Geodesics via the Pontryagin Maximum Principle

We saw how the spatial arc-length parameterization of geodesics in the $\boldsymbol{P}_{curve}$ problem breaks down with the appearance of a cusp. An alternative approach is using the $SE(2)$ arc-length parameterization (denoted with parameter t), which has the following defining characteristic:

$$\xi^2 \dot{x}(t)^2 + \xi^2 \dot{y}(t)^2 + \dot{\theta}(t)^2 = 1 \tag{A.1}$$

As long as $0 \leq t \leq t_{cusp}$ (i.e. we have not reached the first cusp, or there are no cusps) transforming from $SE(2)$ arc-length to spatial arc-length is straightforward:

$$\frac{ds}{dt} = \cos\theta \ \dot{x} + \sin\theta \ \dot{y} = u^1 = \frac{1}{\sqrt{\kappa^2 + 1}} \tag{A.2}$$

And since the transformation is a continuous bijection (under our assumption that no cusps appear on the spatial projections of our geodesics) we can invert this relationship:

$$\frac{dt}{ds} = \left(\frac{ds}{dt}\right)^{-1} \tag{A.3}$$

Taking this into account we can have a look at the canonical equations produced by P.M.P. (See [1, Appendix D], we omit the boundary conditions and set $\xi = 1$):

$$\begin{cases} u^1(\tau) = \cos\theta \ \dot{x} + \sin\theta \ \dot{y} = \lambda_1(t) \\ u^2(\tau) = -\sin\theta \ \dot{x} + \cos\theta \ \dot{y} = 0 \\ u^3(\tau) = \lambda_3(t) \\ \\ \dot{\lambda}_1(t) = \lambda_2(\tau)\lambda_3(t) \\ \dot{\lambda}_2(t) = -\lambda_1(\tau)\lambda_3(t) \\ \dot{\lambda}_3(t) = -\lambda_1(\tau)\lambda_2(t) \end{cases} \tag{A.4}$$

We now look at the effect of reparameterization to spatial arc-length on the momentum components λ_i:

$$\begin{cases} \frac{d\lambda_1}{ds} = \frac{dt}{ds}\frac{d\lambda_1}{dt} = \frac{\lambda_2\lambda_3}{\lambda_1} \\ \frac{d\lambda_2}{ds} = \frac{dt}{ds}\frac{d\lambda_2}{ds} = -\frac{\lambda_1\lambda_3}{\lambda_1} = -\lambda_3 \\ \frac{d\lambda_3}{ds} = \frac{dt}{ds}\frac{\lambda_3}{d\tau} = -\frac{\lambda_1\lambda_2}{\lambda_1} = -\lambda_2 \end{cases} \tag{A.5}$$

Which yields:

$$\Rightarrow \frac{d^2\lambda_3}{ds^2} = -\frac{d\lambda_2}{ds} = \lambda_3 \tag{A.6}$$

Since $\lambda_3 = u^3 = \frac{d\theta}{dt}$ we have:

$$\frac{d\theta}{dt} = \frac{d\theta}{ds} \cdot \frac{ds}{dt} = \frac{\kappa}{\sqrt{\kappa^2 + 1}} = z \tag{A.7}$$

We conclude that (A.6) is equivalent to $\ddot{z} = z$. We can conclude that the P.M.P. scheme applied to $SE(2)$ indeed yields the Euler-Lagrange scheme in $\mathbb{R}^2$ in so far as the spatial arc-length parameterization exists. This new scheme however has the benefit of not being so constricted since the $SE(2)$ arc-length parameterization can exists in and after cusps.

Appendix B

CUDA Implementation

In the process of experimenting with speeding up Orientation Score Transformations using *Mathematica*'s *CUDALink* functions we stumbled on some limitations that precluded us from using convolution kernels of sufficient size. Looking into the details we noted that the limitation was due to *Mathematica* and not *CUDA* and that we could make our own implementation to support arbitrarily large kernel sizes.

Our implementation was guided by [13], which detailes an efficient FFT-based convolution algorithm suitable for GPUs. Figure B.1 detailes the data-flow of our implementation. Essentially we transform both the source image and the wavelet-stack to the Fourier domain and then perform pixel-wise complex multiplications before we transform back to the spatial domain.

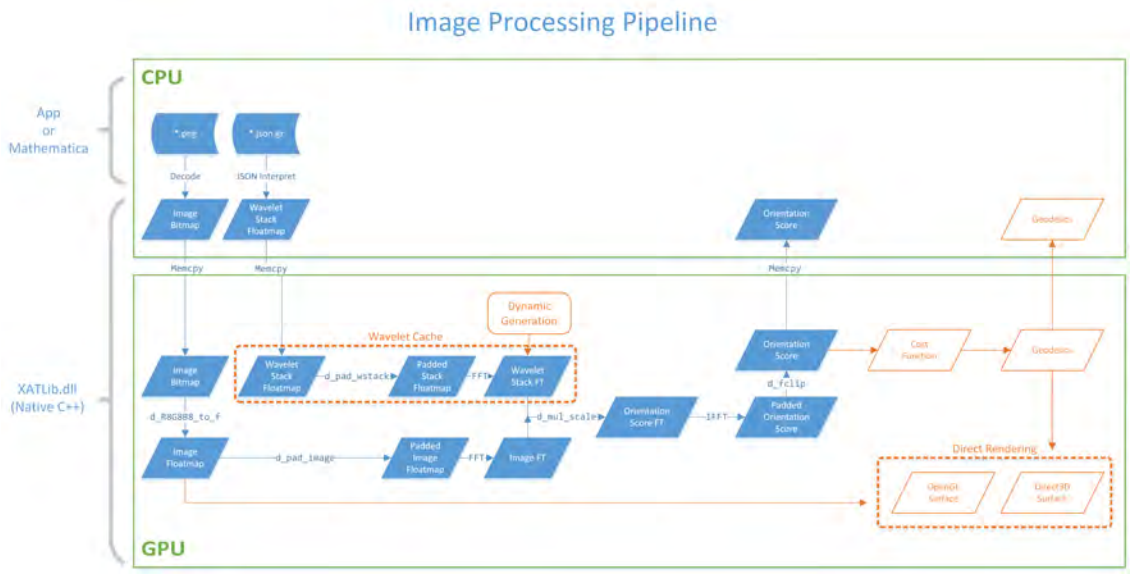


Figure B.1: Image Processing Pipeline (blue: implemented, orange: wishlist)

Our implementation consisted of a DLL that we called from Mathematica using *NETLink* as translation-layer. While we haven't had the opportunity for optimization or detailed timings, we summarize the observed speedup in table B.1.

CPU	GPU	GPU speedup vs. CPU
i7-4700MQ @ 2.40 GHz	Quadro K610M	4x
i7-4790K @ 4.60 GHz	GeForce GTX 1070	10x

Table B.1: Speedup of OS transformations on GPU

List of Figures

List of Tables

Bibliography

[1] R. Duits, U. Boscain, F. Rossi, and Y. Sachkov. Association fields via cuspless sub-riemannian geodesics in SE(2). *Journal of Mathematical Imaging and Vision*, 49(2):384–417, 2014.

[2] Ugo Boscain, Remco Duits, Francesco Rossi, and Yuri Sachkov. Curve cuspless reconstruction via sub-Riemannian geometry. *ESAIM. Control, Optimisation and Calculus of Variations*, 20(3):748–770, 2014.

[3] Gonzalo Sanguinetti, Erik Bekkers, Remco Duits, Michiel H J Janssen, Alexey Mashtakov, and Jean Marie Mirebeau. Sub-riemannian fast marching in SE(2). *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9423(335555):366–374, 2015.

[4] Remco Duits, Stephan P. L. Meesters, Jean-Marie Mirebeau, and Jorg M. Portegies. Optimal Paths for Variants of the 2D and 3D Reeds-Shepp Car with Applications in Image Analysis. 2016.

[5] E. J. Bekkers, R. Duits, A. Mashtakov, and Yu. Sachkov. Vessel Tracking via Sub-Riemannian Geodesics on $\mathbb{R}^2 \times P^{1}$. *Retrieved from http://arxiv.org/abs/1704.04192*, (335555), 2017.

[6] John N. Tsitsiklis. Globally Optimal Trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995.

[7] Jean-Marie Mirebeau. Anisotropic Fast-Marching on Cartesian Grids Using Lattice Basis Reduction. *SIAM J, Numer. Anal.*, 52(4):1573–1599, 2014.

[8] Remco Duits. *Perceptual Organization in Image Analysis*. Number 2003. 2005.

[9] Remco Duits and Erik Franken. Line enhancement and completion via linear left invariant scale spaces on SE(2). *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5567 LNCS(2):795–807, 2009.

[10] Erik J. Bekkers. *Retinal image analysis using sub-Riemannian geometry in SE (2)*. PhD thesis, 2017.

[11] Jiong Zhang, Behdad Dashtbozorg, Erik Bekkers, Josien P.W. Pluim, Remco Duits, and Bart M. Ter Haar Romeny. Robust Retinal Vessel Segmentation via Locally Adaptive Derivative Frames in Orientation Scores. *IEEE Transactions on Medical Imaging*, 35(12):2631–2644, 2016.

[12] T. Calinski and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics - Theory and Methods*, 3(1):1–27, 1974.

[13] Victor Podlozhnyuk. FFT-based 2D convolution. *NVIDIA*, (June), 2007.