Manifolds in Image Science and Visualization

Anders Brun
Cover illustration: A Möbius strip. It is a non-orientable compact manifold with a boundary, discovered independently by August Ferdinand Möbius and Johann Benedict Listing in 1858. It is the canonical example of a one-sided surface, and can be constructed by joining the ends of a strip of paper with a single half-twist. The set of all unordered pairs of line orientations in the plane, $\mathbb{R}^2$, has the topology of a Möbius strip, making this manifold useful for certain tasks in image analysis.

**Manifolds in Image Science and Visualization**

Copyright © 2007 Anders Brun

*Department of Biomedical Engineering*
*Linköpings universitet*
*SE-58185 Linköping, Sweden*

ISBN 978-91-85715-02-2 ISSN 0345-7524

Printed in Linköping, Sweden by UniTryck AB 2007
Alice laughed. “There’s no use trying,” she said, “one can’t believe impossible things.”
“I daresay you haven’t had much practice,” said the Queen.
“When I was your age, I always did it for half-an-hour a day. Why, sometimes I’ve believed as many as six impossible things before breakfast.”

Lewis Carroll, Through the Looking Glass (1871).
Abstract

A Riemannian manifold is a mathematical concept that generalizes curved surfaces to higher dimensions, giving a precise meaning to concepts like angle, length, area, volume and curvature. The sphere gives a glimpse of the characteristics of a non-flat geometry. On the sphere, the shortest path between two points – a geodesic – is along a great circle. Different from Euclidean space, the angle sum of geodesic triangles on the sphere is always larger than 180 degrees.

Sometimes such curved spaces naturally describe signals and data found in applied research. This dissertation presents basic research and tools for the analysis, processing and visualization of such manifold-valued data, with a particular emphasis on future applications in medical imaging and visualization.

Two-dimensional manifolds, i.e. surfaces, enter naturally into the geometric modeling of anatomical entities, such as the human brain cortex and the colon. In advanced algorithms for processing of images obtained from computed tomography (CT) and ultrasound imaging (US), images themselves and derived local structure tensor fields may be interpreted as two- or three-dimensional manifolds. In diffusion tensor magnetic resonance imaging (DT-MRI), the natural description of diffusion in the human body is a second-order tensor field. This tensor field can be related to the metric of a manifold. A final example is the analysis of shape variations of anatomical entities, e.g. the lateral ventricles in the brain, within a population by describing the set of all possible shapes as a manifold.

Works presented in this dissertation include: A probabilistic interpretation of intrinsic and extrinsic means in manifolds; A Bayesian approach to filtering of vector data, removing noise from sampled manifolds and signals; Principles for the storage of tensor field data and learning a natural metric for empirical data.

The main contribution is a novel class of algorithms called LogMaps, for the numerical estimation of $\log p(x)$ from empirical data sampled from a low-dimensional manifold or geometric model embedded in Euclidean space. The $\log p(x)$ function has been used extensively in the literature for processing data in manifolds, including applications in medical imaging such as shape analysis. However, previous approaches have been limited to manifolds where closed form expressions of $\log p(x)$ are known. The introduction of the LogMap framework allows for a generalization of the previous methods. The LogMap framework is also applied to other applications, including texture mapping, tensor field visualization, medial locus estimation and exploratory data analysis.
Populärvetenskaplig sammanfattning

En Riemannmångfald är ett matematiskt begrepp som generaliserar krökta ytor till högre dimensioner och ger mening åt begrepp som vinkel, längd, area, volym och kurvatur i sådana krökta rum. Exempel på konsekvenser av en krökt geometri fästes genom att betrakta sfären, där den kortaste vägen mellan två punkter — en geodet — går längs en storcirkel. Till skillnad från platta Euklidiska rum så är vinkelsumman av geodetiska trianglar på sfären alltid större än 180 grader.

Signaler och data inom tillämpad forskning kan ibland beskrivas naturligt av sådana krökta rum. Denna avhandling presenterar grundforskning och verktyg för att analysera, behandla och visualisera sådan mångfaldsvärd data, med ett speciellt fokus på framtida tillämpningar inom medicinsk bildvetenskap och visualisering.

Tvådimensionella mångfald, alltså ytor, är naturliga för att beskriva geometriska modeller av organ i kroppen, till exempel hjärnburken och tjocktarmen. I avancerad bildbehandling av bilder från datortomografi (CT) och ultraljud (US), kan bilderna sälja och den lokala statistiken i form av strukturtensorfältet tolkas som två- och tre-dimensionella mångfald. I diffusionstensor magnetresonanstomografi (DT-MRI) så beskrivs diffusion i människokroppen med hjälp av ett andra ordningens tensorfält, som kan tolkas som metriken på en mångfald. Slutligen så kan variationer i formen av anatomiska objekt, till exempel de laterala ventrikelfarna i hjärnan, analyseras inom en population genom att beskriva mängden av alla möjliga former med en mångfald.

I denna avhandling presenteras resultat om: Probabilistisk tolkning av intrinsiska och extrinsiska medelvärden i mångfald. En Bayesiansk metod för filtrering av vektor-data, som tar bort brus från samplade mångfald och signaler. Principer för att lagra tensorfält och hur man lära sig en naturlig metrik för empiriska data.

Det viktigaste bidraget är en ny klass av algoritmer kallade LogMaps, som numeriskt skattar \( \log_p(x) \) från empiriska data samplade från en lågdimensionell abstrakt mångfald eller en geometrisk modell i ett Euklidiskt rum. Funktionen \( \log_p(x) \) har använts rikligt inom tidigare forskning om databehandling i mångfald, vilket inkluderar tillämpningar inom medicinsk bildvetenskap såsom formanalys. Tidigare metoder har dock varit begränsade till mångfald där slutna uttryck för \( \log_p(x) \) har funnits. Introduktionen av LogMaps gör det därför möjligt att generalisera tidigare metoder. Resultat presenteras även för att använda LogMaps till texturmappning, visualisering av tensor-fält, skattning av skelett och för att utforska empiriska data.
It all started the 26th of August, 2004. For some time I had been fascinated by the simple fact that the difference between two squared distance functions, e.g. the distance from a point in the plane, $\mathbb{R}^2$, or on the line of real numbers, $\mathbb{R}$, was an affine function. For instance, the squared distance to the point 3 on the line of real numbers, minus the squared distance to 5, is

$$(x - 3)^2 - (x - 5)^2 = 4x - 16.$$ 

This is an affine function, since it has one term that is linear in $x$ and one term that is constant. I cannot explain why I persisted on thinking of this simple relation – it is not exactly a suitable topic for a dissertation or even a scientific paper. Nevertheless, my curiosity led me to ask myself what would happen if tried this for distances on a curved surface or a circle instead of a plane or a line. I decided to try it for the unit circle. In Fig. 1 the squared distance functions for some points on the unit circle are shown, parameterized by $x \in [0, 2\pi]$. All squared distance functions have a sharp cusp, located at the opposite side of the circle relative to the point of reference. At this cusp, there exist two shortest paths along the circle to the point of reference. On the unit circle, the distance between two points is just the angle between the points, measured in radians. It is a simple example of geodesic distance, the length of the shortest path between two points in a manifold.

The difference between squared distance functions can also be seen in Fig. 1. For points far apart, the difference function has the shape of a triangle wave. When the two reference points are close however, in the figure positioned at 1 and 1.1, the difference function is affine and almost linear for most of the interval $[0, 2\pi]$, except between the points where the squared distance functions have cusps. I did not fully understand these results at the time being, but I was encouraged to try this example on a curved surface as well.

To make my next experiment a bit more interesting, I decided to not only try out squared distances on a curved surface, but also to try to use estimated geodesic distances. In many applications that interested me at the time, it was difficult to know the exact geodesic distance between a pair of points in a manifold, because neither the manifold nor the distance function was known from a closed expression. The only thing that was known was a set of points in $\mathbb{R}^N$ sampled from the curved surface or manifold. In a relatively recent work on so-called “manifold learning”, geodesic distances were estimated numerically from samples using Edsger W. Dijkstra’s algorithm for shortest paths in graphs.
To give the reader a snapshot of the everyday life of a PhD student, I have simply included an almost exact replica of the actual code that I used this day to generate Fig. 1. The code for Dijkstra’s algorithm is replaced by some “magic” to make the code self-contained, i.e. without any references to other functions.

The plots seen in Fig. 2 were obtained by running the code in Alg. 1. They showed that the half-sphere has been flattened, mapping geodesic curves from the special points close to the grey dot to lines in the plane. Disregarding some scaling issues, this was in essence the $\log_p(x)$ map, well known in differential geometry and known as the Azimuthal Equidistant Projection in cartography. At the time being, I had no idea that this mapping actually had a name. However, it was beyond doubt that it could be used for non-linear dimension reduction.

Some days later I tried to use the same method to perform dimension reduction on other data, small image patches, that I believed might live on a surface or generally a manifold embedded in a high-dimensional space. It turned out that it was indeed possible. In addition, by selecting patches with oriented patterns of different phase, which you can read more about in chapter 5, I empirically discovered the Klein bottle topology of local phase and orientation in images.

– It was a great week!
Algorithm 1 Authentic MATLAB code for the first LogMap experiments.

```matlab
% Make N samples from a half-sphere.
N = 2000;
X = randn(N,3);
X = X./repmat(sqrt(sum(X.^2),2),[1 3]);
X(:,3) = X(:,3) .* sign(X(:,3));

% Add three "special points" close to the top.
X(N,:) = [0 0 1];
X(N-1,:) = [0.1 0 0.995];
X(N-2,:) = [0 0.1 0.995];

% Plot the point cloud in top left figure.
subplot(2,2,1); scatter3(X(:,1),X(:,2),X(:,3),10);
axis equal;

% Estimate geodesic distances between special and
% other points. With a little exp-log magic!
G = zeros(N,N);
for k = 1:3
    G = G + (X(:,k)*ones(1,N)-ones(N,1)*X(:,k)')^2;
end
G = sqrt(G); %Euclidean distance between all points
G(G>0.3) = inf; GD = ones(N,3)*inf;
GD(N,3) = 0; GD(N-1,2) = 0; GD(N-2,1) = 0;
for k = 1:10
    GD = log(exp(-400*G)*exp(-400*GD))/-400;
end

% Calculate the mapping using "special points".
V = 0.5*10*(GD(:,3).^2-GD(:,2).^2);
V(:,2) = 0.5*10*(GD(:,3).^2-GD(:,1).^2);

% Plot the point cloud after mapping in the top
% right figure.
subplot(2,2,2); scatter(V(:,1),V(:,2),4); axis equal;

% Compare radial distance after the mapping
% with known geodesic distance on the half-sphere.
EL = sqrt(V(:,1).^2 + V(:,2).^2);
RL = acos(X(:,3));
subplot(2,2,3); scatter(EL,RL,4);

% Compare angular argument before and after the mapping
EA = angle(V(:,1) + V(:,2)*i);
RA = angle(X(:,1) + X(:,2)*i);
subplot(2,2,4); scatter(EA,RA,1); axis([-pi pi -pi pi]);
```
Figure 2: The first LogMap experiment. Top-Left: Points on a half-sphere embedded in $\mathbb{R}^3$. Grey dots indicate the three special points. Top-Right: Points mapped by the algorithm to $\mathbb{R}^2$. Bottom-Left: A comparison of true and estimated geodesic distances. Bottom-Right: A comparison of true and estimated angles. See the code in Alg. 1 for further explanations.
Acknowledgements

The work presented in this thesis would have been difficult without the support of a number of people.

I would like to thank my supervisor Professor Hans Knutsson for excellent guidance into the unknown. His many contributions to image processing led me to the Medical Informatics group in the first place and it has been a fun and inspiring environment during the past years.

My co-supervisor Associate Professor Carl-Fredrik Westin, from the Laboratory of Mathematics in Imaging at Brigham and Women’s Hospital and Harvard Medical School, provided a great introduction to the field of Diffusion Tensor MRI and guided me around at various bars and events in Boston.

My other co-supervisor Associate Professor Magnus Herberthson, from the Department of Mathematics at Linköping University, taught me everything else I know about the mathematics of tensors, manifolds, normed spaces and the obvious connections to ice hockey hooliganism.

Former colleagues in Boston, in particular Professor Martha E. Shenton, Professor Hae-Jeong Park, Assistant Professor Marek Kubicki, Dr. Steven Haker and Dr. Raúl San-José Estépar, Dr. Karl Krissian, Dr. Gordon Kindlmann and Dr. Lauren O’Donnell.

The Similar WP10 and Tensor Grand Challenge members, in particular Dr. Marcos Martin-Fernandez, Dr. Burak Acar, Emma Munoz-Moreno, Dr. Leila Cammoun and Dario Sosa.

The MOVIII fMRI demonstrator crew: Dr. Jacob Roll, Henrik Ohlsson, Dr. Mats Andersson, Professor Hans Knutsson, Joakim Rydell, Professor Anders Ynnerman and Professor Lennart Ljung.

Mats Björnemo, now I can tell you what a tensor really is.

Dr. Hans Rullgård, for transforming intuition into amazing mathematics.

Dr. Lisa Falco, for the many philosophical and DT-MRI related discussions over the years.

Ola Nilsson, who made a remarkable effort to increase the accuracy of distance transforms in mesh geometries, with a never-ending enthusiasm.

Dr. Thomas Schön, for adventures both in $SO(3)$ and at the shooting range.

John Wilander, for philosophical discussions on life, PhD studies, politics and dating.
Friends and colleagues at LinTek, StuFF and Consensus, in particular Carina Andersson.

My colleagues at the Department of Biomedical Engineering at Linköping University, in particular the Medical Informatics group and my closest co-workers I met during the years: Dr. Mats Andersson, Associate Professor Magnus Borga, Dr. Gunnar Farnebäck, Dr. Ola Friman, Johan Wiklund, Andreas Wrangsjö, Nina Eriksson-Bylund, Dr. Kenneth Andersson, Johanna Pettersson, Thord Andersson, Dr. Joakim Rydell, Björn Svensson and Andreas Sigfridsson.

Friends in Linköping and around the world. In particular selected members of SNY: Anna-Karin Sundquist, Dr. Michael Öster, Rikard Andersson, Katrin Karlsson and Jonas Gustavsson.

The fabulous Persson family, in particular Annica and Lars who at times have provided something close to a second home for me. And of course also Daniel, Anna, Ingrid, Saga, Matte and Dr. Selma!

My relatives in Sala, Västerås, Göteborg and Helsingborg, for their love and support during a difficult year. I cherish the memory of Ingrid, Håkan and Anita.

My dearest parents: My father Paul. My mother Anita, who is no longer with us. Thank you for everything you have given to me.

Most of all, my love Elise. You make me happy. And to you I dedicate my life.

I gratefully acknowledge the permission from IEEE\(^1\) and Springer\(^2\) to reproduce previously published work with minor changes in chapter 10 and 11.

Finally, I would like to acknowledge the financial support I have received from various sources: The European Union (FP6) funded Similar Network of Excellence; the Center for Medical Image Science and Visualization (CMIV and SMIV) at Linköping University; the center for Non-Invasive Medical Measurements (NIMED) funded by the Swedish Governmental Agency for Innovation Systems (VINNOVA); the center for Modelling, Visualization and Information Integration (MOVIII) funded by the Swedish Foundation for Strategic Research (SSF) and the Manifold-Valued Signal Processing project funded by the Swedish Research Council (VR).

\(^1\)Chapter 10: Intrinsic and Extrinsic Means on the Circle – a Maximum Likelihood Interpretation, by A. Brun, C.-F. Westin, M. Herberthson, H. Knutsson, Proceedings of IEEE International Conference on Acoustics, Speech, & Signal Processing, Honolulu, Hawaii, USA April 2007. This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of Linköpings universitet’s products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org.

# Table of Contents

1 Introduction  
1.1 Motivations  
1.2 Potential impact  
1.3 Dissertation overview  
1.4 Contributions  
1.5 Publications  
1.6 Abbreviations  
1.7 Mathematical Notation  

2 Mathematics  
2.1 Linear algebra  
2.1.1 Vector spaces  
2.1.2 Linear maps  
2.1.3 The dual vector space  
2.1.4 The Einstein summation convention  
2.1.5 Coordinate changes  
2.1.6 Inner products and metrics  
2.2 Tensors  
2.2.1 Outer products  
2.2.2 Cartesian tensors  
2.2.3 Index gymnastics  
2.3 Manifolds  
2.3.1 Charts and atlases  
2.3.2 The tangent space  
2.3.3 Geodesic length and distance  
2.3.4 Further reading  

3 Dimension reduction and manifold learning  
3.1 Machine learning  
3.1.1 Dimensionality reduction  
3.1.2 Manifold learning  
3.1.3 Laplacian eigenmaps  
3.1.4 Isomap — isometric feature mapping  
3.1.5 A brief historical timeline  

4 Diffusion tensor MRI  
4.1 Diffusion imaging
Table of Contents

4.1 Diffusion .................................................. 26
4.1.1 Diffusion ............................................. 26
4.1.2 Estimating diffusion tensors ......................... 27
4.1.3 Diffusion in the human brain ....................... 27
4.1.4 Applications of DT-MRI ............................. 28
4.2 Processing diffusion tensor data ....................... 29
4.2.1 Scalar invariants .................................... 29
4.2.2 Fiber tracking ...................................... 32
4.2.3 Fiber tract connectivity ........................... 32
4.2.4 Segmentation of white matter ..................... 34
4.3 Visualization of streamline data ...................... 34
4.3.1 Local and global features in DT-MRI ............. 34
4.3.2 Visualization of fiber tract connectivity .......... 35

5 Empirical LogMaps .......................................... 39
5.1 Introduction ............................................. 39
5.2 Related work ............................................ 40
5.2.1 Programming on manifolds ....................... 41
5.2.2 Previous work on Riemannian normal coordinates 41
5.3 The LogMap algorithm ................................. 42
5.4 Mathematical properties of RNC and LogMaps ....... 46
5.4.1 The LogMap formula ............................... 46
5.4.2 On the optimality of LogMaps .................... 47
5.5 Experiments ............................................. 48
5.5.1 The Swiss roll ...................................... 48
5.5.2 The torus ........................................... 48
5.5.3 Local phase ....................................... 48
5.5.4 Blob-shapes ....................................... 48
5.5.5 Conclusion ......................................... 54

6 LogMap texture mapping ................................. 57
6.1 Introduction ............................................. 57
6.2 Previous work .......................................... 57
6.3 The LogMap method .................................. 59
6.4 Computing geodesic distance ......................... 59
6.5 Experiments ............................................. 61
6.5.1 The Stanford bunny ............................... 61
6.5.2 Plane with a bump ................................ 62
6.5.3 A model problem .................................. 62
6.6 Conclusions and future work ....................... 64

7 Estimating skeletons from LogMap ..................... 67
7.1 Algorithm .............................................. 67
7.2 Experiments ............................................ 69
7.3 Conclusion ............................................. 70
Table of Contents

8 Geodesic glyph warping
  8.1 Introduction .................................................. 75
  8.2 Related work .................................................... 76
  8.3 Index notation .................................................. 79
  8.4 The metric and metric spheres ................................. 81
  8.5 The geodesic equation and geodesic spheres ................. 82
  8.6 The exponential map and Riemannian normal coordinates .... 84
  8.7 Solving the geodesic equation ................................ 85
  8.8 Geodesic spheres and warped coordinate systems ........... 86
  8.9 The logarithmic map ............................................ 86
  8.10 Experiments .................................................... 87
  8.11 Conclusion ..................................................... 90

9 Natural metrics for parameterized image manifolds
  9.1 Introduction .................................................... 91
  9.2 Related work .................................................... 93
  9.3 A model for image manifolds .................................. 94
  9.4 An experiment: Intrinsic geometry in DWI .................... 97
  9.5 Conclusions ..................................................... 101

10 Intrinsic and extrinsic means
  10.1 Introduction ................................................... 103
    10.1.1 The intrinsic mean ....................................... 103
    10.1.2 The extrinsic mean ...................................... 104
  10.2 Modeling noise by Brownian motion .......................... 105
    10.2.1 Means as ML estimates in $\mathbb{R}^n$ .................... 105
    10.2.2 Intrinsic means as ML estimates in $S^1$ ............... 106
    10.2.3 Extrinsic means as ML estimates in $S^1$ ............... 107
  10.3 Experiments ................................................... 108
  10.4 Discussion ................................................... 112

11 Bayesian feature space filtering
  11.1 Introduction ................................................... 117
  11.2 Previous work ................................................ 118
  11.3 The Bayesian method .......................................... 118
    11.3.1 Noise models ............................................ 118
    11.3.2 Signal models for images ................................ 119
    11.3.3 Signal models for N-D data sets ....................... 119
    11.3.4 Estimation ............................................... 119
  11.4 Importance sampling ......................................... 120
    11.4.1 Proper samples .......................................... 120
    11.4.2 Importance sampling .................................... 121
  11.5 Implementation ............................................... 121
    11.5.1 Vector-valued images .................................... 122
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.5.2</td>
<td>Unordered N-D data</td>
<td>123</td>
</tr>
<tr>
<td>11.6</td>
<td>Experiments</td>
<td>123</td>
</tr>
<tr>
<td>11.6.1</td>
<td>Scalar signals</td>
<td>123</td>
</tr>
<tr>
<td>11.6.2</td>
<td>Vector-valued signals</td>
<td>123</td>
</tr>
<tr>
<td>11.6.3</td>
<td>Unordered N-D data</td>
<td>127</td>
</tr>
<tr>
<td>11.7</td>
<td>Conclusion</td>
<td>130</td>
</tr>
<tr>
<td>12</td>
<td>Storing regularly sampled tensor charts</td>
<td>131</td>
</tr>
<tr>
<td>12.1</td>
<td>Introduction</td>
<td>131</td>
</tr>
<tr>
<td>12.2</td>
<td>Related work</td>
<td>132</td>
</tr>
<tr>
<td>12.3</td>
<td>Geometric arrays</td>
<td>133</td>
</tr>
<tr>
<td>12.4</td>
<td>Scalar array data storage</td>
<td>134</td>
</tr>
<tr>
<td>12.5</td>
<td>Tensor array data storage</td>
<td>135</td>
</tr>
<tr>
<td>12.6</td>
<td>The tensor array core</td>
<td>136</td>
</tr>
<tr>
<td>12.6.1</td>
<td>Storing array data</td>
<td>137</td>
</tr>
<tr>
<td>12.7</td>
<td>Examples</td>
<td>138</td>
</tr>
<tr>
<td>12.8</td>
<td>Discussion</td>
<td>139</td>
</tr>
<tr>
<td>13</td>
<td>Summary and outlook</td>
<td>141</td>
</tr>
<tr>
<td>13.1</td>
<td>Future Research</td>
<td>141</td>
</tr>
</tbody>
</table>
### List of Figures

1. Squared distance function on the unit circle ............... x
2. The first LogMap experiment (results) .................. xii
2.1 Covariant and contravariant vectors .................. 11
2.2 Coordinate changes in physics, an example .......... 12
2.3 Charting a manifold .................................. 15
3.1 Examples of immersed and embedded manifolds .......... 18
3.2 A linear model ......................................... 19
3.3 A non-linear model ..................................... 20
3.4 A graph-based model .................................... 21
4.1 Diffusion-Weighted MRI scans .......................... 28
4.2 Typical glyphs for linear, planar and spherical tensors .. 31
4.3 Dissection of a brain revealing the structure of white matter .. 33
4.4 Dissection of a brain revealing the structure of white matter .. 33
4.5 Axial images of a brain derived from DT-MRI .......... 36
4.6 Axial images of a brain derived from DT-MRI (details) .. 36
4.7 Streamtube coloring of DT-MRI fiber traces ............ 36
5.1 A schematic view of $\exp(x)$, $\log_p(x)$ and $T_p M$ ........ 40
5.2 A schematic view of the LogMap framework ............ 42
5.3 The Swiss roll experiment using LogMaps ............... 50
5.4 A torus experiment using LogMaps ..................... 51
5.5 A map of all image phase/orientations for patches ....... 52
5.6 Discovering the Klein bottle in small image patches .... 53
5.7 Examples of LogMaps in the Klein bottle of phase/orientation .. 54
5.8 The Klein bottle is not a globally symmetric manifold .... 55
5.9 A blob image experiment using LogMaps ............... 56
6.1 A schematic view of LogMap for surfaces ............... 58
6.2 LogMap texture mapping of the Stanford bunny .......... 63
6.3 LogMap texture mapping of the Stanford bunny (cut locus) ... 64
6.4 LogMap texture mapping of a plane with a bump ........ 65
6.5 LogMap texture mapping of a model sphere ............. 65
6.6 A comparison of Reimers algorithm versus exact distances .. 66
7.1 A schematic view on medial locus estimation using LogMap . . 68
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.2</td>
<td>Closed and open curves in the plane</td>
<td>71</td>
</tr>
<tr>
<td>7.3</td>
<td>LogMap coordinates for closed and open curves</td>
<td>72</td>
</tr>
<tr>
<td>7.4</td>
<td>Estimated skeleton for closed and open curves</td>
<td>73</td>
</tr>
<tr>
<td>8.1</td>
<td>The Tissot indicatrix</td>
<td>77</td>
</tr>
<tr>
<td>8.2</td>
<td>Geodesic sphere glyphs on a half-sphere and a cone</td>
<td>78</td>
</tr>
<tr>
<td>8.3</td>
<td>Coordinate basis vectors in $\mathbb{R}^2$ derived for some metric $g_{ij}$</td>
<td>82</td>
</tr>
<tr>
<td>8.4</td>
<td>A schematic view of the $\exp_p$ and $\log_p$</td>
<td>84</td>
</tr>
<tr>
<td>8.5</td>
<td>Examples of geodesic normal coordinates</td>
<td>88</td>
</tr>
<tr>
<td>8.6</td>
<td>Examples of geodesic sphere glyphs</td>
<td>88</td>
</tr>
<tr>
<td>8.7</td>
<td>Examples of geodesically warped box glyphs</td>
<td>89</td>
</tr>
<tr>
<td>8.8</td>
<td>Examples of other geodesically warped glyphs</td>
<td>89</td>
</tr>
<tr>
<td>9.1</td>
<td>Difficulties in estimation a metric for a point cloud</td>
<td>92</td>
</tr>
<tr>
<td>9.2</td>
<td>Estimating a metric in time series data</td>
<td>93</td>
</tr>
<tr>
<td>9.3</td>
<td>An illustration of sample- and index space</td>
<td>95</td>
</tr>
<tr>
<td>9.4</td>
<td>The remapping of the b-value</td>
<td>98</td>
</tr>
<tr>
<td>9.5</td>
<td>A natural metric experiment (a)</td>
<td>99</td>
</tr>
<tr>
<td>9.6</td>
<td>A natural metric experiment (b)</td>
<td>99</td>
</tr>
<tr>
<td>9.7</td>
<td>A natural metric experiment (c)</td>
<td>100</td>
</tr>
<tr>
<td>9.8</td>
<td>A natural metric experiment (d)</td>
<td>100</td>
</tr>
<tr>
<td>10.1</td>
<td>The difference between intrinsic an extrinsic means</td>
<td>104</td>
</tr>
<tr>
<td>10.2</td>
<td>Diffusion means on the circle (short time kernel)</td>
<td>109</td>
</tr>
<tr>
<td>10.3</td>
<td>Diffusion means on the circle (medium time kernel)</td>
<td>110</td>
</tr>
<tr>
<td>10.4</td>
<td>Diffusion means on the circle (long time kernel)</td>
<td>111</td>
</tr>
<tr>
<td>10.5</td>
<td>Comparing extrinsic and intrinsic means, 3 samples</td>
<td>114</td>
</tr>
<tr>
<td>10.6</td>
<td>Comparing extrinsic and intrinsic means, 3 samples</td>
<td>114</td>
</tr>
<tr>
<td>10.7</td>
<td>Comparing extrinsic and intrinsic means, 100 samples</td>
<td>115</td>
</tr>
<tr>
<td>10.8</td>
<td>Comparing extrinsic and intrinsic means, 3 samples</td>
<td>115</td>
</tr>
<tr>
<td>11.1</td>
<td>How trial distributions affect importance sampling</td>
<td>120</td>
</tr>
<tr>
<td>11.2</td>
<td>Filtering a 1-D scalar signal</td>
<td>124</td>
</tr>
<tr>
<td>11.3</td>
<td>Filtering a noisy 2-D scalar image</td>
<td>124</td>
</tr>
<tr>
<td>11.4</td>
<td>Filtering a noisy 2-D RGB image</td>
<td>125</td>
</tr>
<tr>
<td>11.5</td>
<td>Filtering a noisy 2-D RGB image (close up)</td>
<td>126</td>
</tr>
<tr>
<td>11.6</td>
<td>Filtering unordered 2-D data</td>
<td>128</td>
</tr>
<tr>
<td>11.7</td>
<td>Filtering unordered 3-D data</td>
<td>129</td>
</tr>
<tr>
<td>12.1</td>
<td>Canonical layout of array data</td>
<td>133</td>
</tr>
<tr>
<td>12.2</td>
<td>Tensor array transformations (a warning example)</td>
<td>135</td>
</tr>
<tr>
<td>12.3</td>
<td>Tensor array transformations (a correct example)</td>
<td>136</td>
</tr>
<tr>
<td>12.4</td>
<td>Tensor array transformations (a correct example)</td>
<td>136</td>
</tr>
<tr>
<td></td>
<td>Algorithm</td>
<td>Page</td>
</tr>
<tr>
<td>---</td>
<td>-----------------------------------------------------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>1</td>
<td>The first LogMap experiment (MATLAB code)</td>
<td>xi</td>
</tr>
<tr>
<td>2</td>
<td>LogMap: General estimate of $x = \log_p(x)$</td>
<td>44</td>
</tr>
<tr>
<td>3</td>
<td>Classical Multidimensional Scaling (MDS)</td>
<td>44</td>
</tr>
<tr>
<td>4</td>
<td>LogMap, Simplified LogMap Estimation</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>LogMap for triangular surface meshes</td>
<td>60</td>
</tr>
<tr>
<td>6</td>
<td>Euclidian distance computation</td>
<td>62</td>
</tr>
<tr>
<td>7</td>
<td>Gaussian Normal Coordinates for a closed curve $\Gamma$</td>
<td>69</td>
</tr>
</tbody>
</table>
List of Tables

4.1 Typical ADC values found in the human brain . . . . . . . . . . . 28
5.1 Analogies between vector spaces and manifolds . . . . . . . . . . 41
12.1 A table of a minimalistic scalar array format. . . . . . . . . . . 134
12.2 A table of the tensor array core . . . . . . . . . . . . . . . . . . 138
1

Introduction

1.1 Motivations

The work presented in this dissertation was inspired by recent advances in so-called manifold learning and mainly financed by the Manifold-Valued Signal Processing project funded by Vetenskapsrådet (the Swedish Research Council).

The need for methods for high-dimensional data analysis and visualization, both in image science in general and in medical image science in particular, motivates the focus on manifolds. Texture, shape, orientation and many other aspects of data need to be quantified, compared and visualized, and the mathematical theory of smooth Riemannian manifolds provide a natural framework for many such tasks.

The use of manifolds and manifold learning, for image analysis and visualization, is explored from three different views in this dissertation:

**Dimension reduction:** Finding a low-dimensional parameterization of manifold-valued data embedded in a high-dimensional space.

**Data visualization:** Visualization of manifolds and manifold-valued data, using exploratory dimension reduction, texture mapping and tensor glyphs.

**Processing and Storage:** Efficient algorithms for signal processing, such as interpolation, smoothing and filtering of manifolds and manifold-valued data. Standard ways to store and communicate data, in particular tensor fields on manifolds.

1.2 Potential impact

The outcome of this work is a new set of tools to understand and process manifold-valued signals, which may or may not be embedded in a high dimensional space. Increased ability to represent and process features present in medical images, such
as shape, texture and organ orientation, will aid in the development of better diagnoses and increase our ability to make demographical studies using data from the imaging sciences. This is of benefit not only within our field of research, which is medical image analysis, but also for the signal processing community as a whole, where there is a need to visualize, process and communicate manifold data.

1.3 Dissertation overview

The dissertation consists of three parts. The first part (chapters 1–4) is an introduction:

Chapter 2 The reader is introduced to some basic concepts in linear algebra, tensors and smooth Riemannian manifolds.

Chapter 3 An introduction to dimension reduction and manifold learning. Some basic algorithms are described and we give a brief historical time-line of the developments in the field.

Chapter 4 A short introduction to Diffusion Tensor MRI (DT-MRI). Despite a strong focus on basic research in the dissertation, DT-MRI is a recurring theme in several chapters. It is the canonical example of the need for advanced image processing in medicine and it has strong connections to Riemannian manifolds.

The second part (chapters 5–12) consists of new theory and applications:

Chapter 5 introduces empirical LogMaps, a framework for a non-linear dimension reduction which is strongly connected to differential geometry and Riemannian manifolds.

Chapter 6 applies LogMaps to a texture mapping problem in computer graphics. In particular, it demonstrates the behavior of the LogMap algorithm when accurate distance estimates are provided.

Chapter 7 contains notes on how the LogMap method can be used to estimate the medial locus, also known as the skeleton, for objects in the plane. The results in this short chapter are preliminary, but encouraging.

Chapter 8 describes a method to visualize curvature in tensor fields and manifolds. It is based on the exponential map, a close relative to the LogMap. It is a general technique, which can be used to warp any metric tensor glyphs according to the curvature of the metric field.

Chapter 9 discusses the problem of finding a natural metric in image manifolds derived from first principles. The basic assumption is that there exists a manifold in which certain local statistical properties of the data are isotropic.
Chapter 10 compares the intrinsic and extrinsic means on the unit circle. This is a very basic problem related to signal processing in globally symmetric manifolds in general. It provides a warning example, showing that the intrinsic formulation of a mean is not always the best estimator. It also gives statistical meaning to both the intrinsic and extrinsic means on the circle.

Chapter 11 describes a computational framework based on importance sampling and particles, to filter vector-valued data and signals such as images and manifolds.

Chapter 12 describes the principles for a canonical file format for the storage and communication of tensor fields stored as multi-dimensional arrays.

In the final part (chapter 13) the work is summarized and possible future research is discussed:

Chapter 13 contains a summary of the importance of the methods and findings presented in the dissertation, and some possible directions for future research.

1.4 Contributions

To emphasize the novel contributions in the thesis, here is a list:

- The empirical LogMaps presented in chapter 5 is a new kind of manifold learning technique, with a strong connection to differential geometry and with interesting computational aspects. The material originates from (Brun et al., 2005) and (Brun, 2006), but some aspects of the framework has been clarified and new theorems are presented in this dissertation.

- The application of LogMaps to the texture mapping problem is a novel contribution. It is one of the first real-world applications of the LogMap framework and it is based on (Brun et al., 2007a) that was recently submitted. In particular, we provide some results on convergence, testing the performance of LogMaps on a model problem. It should also be noted that Ola Nilsson made major contributions for the coding of the Bunny renderings and the distance estimation algorithms on triangular meshes.

- In chapter 7 a novel approach to the estimation of the medial locus of a closed or open curve in the plane is demonstrated. It can be generalized to curved manifolds with a border, making it interesting to for instance the computer graphics community. In addition to finding the medial locus, it also estimates an interesting coordinate system related to a curve.

- The geodesic glyph warping presented in chapter 8 provides a novel way to visualize curvature in diffusion tensor fields and manifolds, using anisotropic
glyphs that are bent according to the curvature. This method has applications in diffusion tensor MRI and it has previously been submitted as a book chapter (Brun and Knutsson, 2007).

- Chapter 9 on natural metrics for image manifolds introduces a novel statistical model based on random fields, from which a “natural” metric is derived for a manifold of images. This idea is related to the structure tensor approach in image analysis, but it is unique in its interpretation of data as a manifold and by the fact that it averages outer products of gradients in a point and not over a spatial neighborhood.

- The chapter on intrinsic and extrinsic means on the circle is derived from (Brun et al., 2007c) and also include experiments that demonstrate the fact that intrinsic means are inferior to extrinsic means under certain statistical circumstances. To the best of our knowledge, this is a novel interpretation of the extrinsic mean. Previously the extrinsic mean has been seen mostly as an approximation to the intrinsic mean.

- The Bayesian feature space filtering is a novel computational paradigm for filtering based on particles and importance sampling. This chapter is derived from (Brun et al., 2007b) and extends previous work (Wrangsjö, 2004; Wrangsjö et al., 2004) from scalar signals, to vector-valued signals and unordered data.

- Finally the work on a standard for the storage of tensor fields, presented in chapter 12, is the result of a collaborative effort within the Similar Network of Excellence (FP6) in which the author has made the major contribution. It is a novel and minimalist approach to the problem of storage and communication of sampled tensor field data. It has previously been presented at a dedicated tensor workshop (Brun et al., 2006).

1.5 Publications

Most of the chapters in this dissertation build upon research that have been presented previously, been submitted or is in manuscript form. This dissertation is solely based on research where the author, that is me, has substantially contributed to ideas, experiments, illustrations and writing, which in every case amounts to at least half of the total work.

This dissertation is based on the following material:

2. Tensor Glyph Warping – Visualizing Metric Tensor Fields using Riemann-
ian Exponential Maps, A. Brun, H. Knutsson, submitted as book chapter.

3. Riemannian Normal Coordinates from Distance Functions on Triangular
Meshes, A. Brun, O. Nilsson, H. Knutsson, submitted to a conference and
in manuscript for journal publication.

4. A Natural Metric in Image Manifolds, preliminary, in manuscript.

5. Using Importance Sampling for Bayesian Feature Space Filtering, A. Brun,
B. Svensson, C.-F. Westin, M. Herberthson, A. Wrangsjö, H. Knutsson Pro-
ceedings of the 15th Scandinavian conference on image analysis (SCIA’07),
Aalborg, Denmark June 2007. Also in manuscript for journal publication.

6. Intrinsic and Extrinsic Means on the Circle - a Maximum Likelihood Inter-
pretation, A. Brun, C.-F. Westin, M. Herberthson, H. Knutsson Proceedings
of IEEE International Conference on Acoustics, Speech, & Signal Process-
ing, Honolulu, Hawaii, USA April 2007.

7. Similar Tensor Arrays - a Framework for Storage of Tensor Data, A. Brun,
M. Martin-Fernandez, B. Acar, E. Muñoz-Moreno, L. Cammoun, A. Sigfridsson,
D. Sosa-Cabrera, B. Svensson, M. Herberthson, H. Knutsson Similar
NoE Tensor Workshop, Las Palmas, Spain, Technical Report, November
2006.

8. Manifold Learning and Representations for Image Analysis and Visualiza-
tion, A. Brun Lic Thesis March 2006.

9. Fast Manifold Learning Based on Riemannian Normal Coordinates, A. Brun,
C.-F. Westin, M. Herberthson, H. Knutsson SCIA 2005, Joensuu, Fin-
land, June 2005.

Material related to this work but not reviewed in this dissertation

1. Representing pairs of orientations in the plane, M. Herberthson, A. Brun,
H. Knutsson Proceedings of the 15th Scandinavian conference on image
analysis (SCIA’07), Aalborg, Denmark June 2007. A journal version is in
manuscript.

2. Estimation of Non-Cartesian Local Structure Tensor Fields, B. Svensson,
A. Brun, M. Andersson, H. Knutsson Proceedings of the 15th Scandinavian
conference on image analysis (SCIA’07), Aalborg, Denmark June 2007.

3. P-Averages of Diffusion Tensors M. Herberthson, A. Brun, H. Knutsson Proceed-
ing of the SSBA Symposium on Image Analysis, Linköping, Sweden
March 2007

4. A tensor-like representation for averaging, filtering and interpolation of 3-D
object orientation data, A. Brun, C.-F. Westin, S. Haker, H. Knutsson ICIP


### 1.6 Abbreviations

A list of abbreviations used in the thesis.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>Apparent Diffusion Coefficient</td>
</tr>
<tr>
<td>CCA</td>
<td>Canonical Correlation Analysis / Curvilinear Components Analysis</td>
</tr>
<tr>
<td>C-Isomap</td>
<td>Conformal Isomap</td>
</tr>
<tr>
<td>CSF</td>
<td>Cerebrospinal Fluid</td>
</tr>
<tr>
<td>DT-MRI</td>
<td>Diffusion Tensor Magnetic Resonance Imaging</td>
</tr>
<tr>
<td>DWI</td>
<td>Diffusion Weighted Imaging</td>
</tr>
<tr>
<td>EOF</td>
<td>Empirical Orthogonal Functions</td>
</tr>
<tr>
<td>FA</td>
<td>Fractional Anisotropy</td>
</tr>
<tr>
<td>FIR</td>
<td>Finite Impulse Response</td>
</tr>
<tr>
<td>GTM</td>
<td>Generative Topographic Map</td>
</tr>
<tr>
<td>HLLE</td>
<td>Hessian Locally Linear Embedding</td>
</tr>
<tr>
<td>ICA</td>
<td>Independent Components Analysis</td>
</tr>
<tr>
<td>i.i.d.</td>
<td>independent and identically distributed</td>
</tr>
<tr>
<td>Isomap</td>
<td>Isometric Feature Mapping</td>
</tr>
<tr>
<td>KPCA</td>
<td>Kernel Principal Components Analysis</td>
</tr>
<tr>
<td>L-Isomap</td>
<td>Landmark Isomaps</td>
</tr>
<tr>
<td>LE</td>
<td>Laplacian Eigenmaps</td>
</tr>
<tr>
<td>LLE</td>
<td>Locally Linear Embedding</td>
</tr>
<tr>
<td>LSI</td>
<td>Latent Semantic Indexing</td>
</tr>
<tr>
<td>LSDI</td>
<td>Line Scan Diffusion weighted Imaging</td>
</tr>
<tr>
<td>LTSA</td>
<td>Local Tangent Space Alignment</td>
</tr>
<tr>
<td>MDS</td>
<td>Multidimensional Scaling</td>
</tr>
<tr>
<td>MR</td>
<td>Magnetic Resonance</td>
</tr>
<tr>
<td>MRI</td>
<td>Magnetic Resonance Imaging</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Components Analysis</td>
</tr>
<tr>
<td>PDD</td>
<td>Principal Diffusion Direction</td>
</tr>
<tr>
<td>PP</td>
<td>Projection Pursuit</td>
</tr>
<tr>
<td>RGB</td>
<td>Red, Green, Blue</td>
</tr>
<tr>
<td>SOM</td>
<td>Self Organizing Maps</td>
</tr>
</tbody>
</table>
### 1.7 Mathematical Notation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>Unspecified vectors</td>
</tr>
<tr>
<td>(b_i)</td>
<td>A contravariant basis vector</td>
</tr>
<tr>
<td>(b^i)</td>
<td>A covariant basis vector</td>
</tr>
<tr>
<td>(v^i)</td>
<td>(The coordinates of) a contravariant vector</td>
</tr>
<tr>
<td>(w_i)</td>
<td>(The coordinates of) a covariant vector</td>
</tr>
<tr>
<td>(g_{ij})</td>
<td>(The components of) the metric tensor</td>
</tr>
<tr>
<td>(M)</td>
<td>A manifold</td>
</tr>
<tr>
<td>(TM)</td>
<td>The tangent bundle of (M)</td>
</tr>
<tr>
<td>(T^*M)</td>
<td>The cotangent bundle of (M)</td>
</tr>
<tr>
<td>(T_pM)</td>
<td>The tangent space of (M) at the point (p)</td>
</tr>
<tr>
<td>(T_p^*M)</td>
<td>The cotangent space of (M) at the point (p)</td>
</tr>
<tr>
<td>(V^*)</td>
<td>The dual vector space of a vector space (V)</td>
</tr>
<tr>
<td>(\dim V)</td>
<td>The dimensionality of (V)</td>
</tr>
<tr>
<td>(\hat{e}_i)</td>
<td>A unit basis vector in (T_pM)</td>
</tr>
<tr>
<td>(g)</td>
<td>A gradient vector in (T_p^*M)</td>
</tr>
<tr>
<td>(X)</td>
<td>A set of data points on (M) embedded in (\mathbb{R}^N)</td>
</tr>
<tr>
<td>(x, y)</td>
<td>Points on (M) embedded in (\mathbb{R}^N)</td>
</tr>
<tr>
<td>(p)</td>
<td>A point on a manifold</td>
</tr>
<tr>
<td>(B_r(p))</td>
<td>A ball of (p) with radius (r) in a set</td>
</tr>
<tr>
<td>(N(p))</td>
<td>A neighborhood of (p) in a set</td>
</tr>
<tr>
<td>(H(t))</td>
<td>A curve along a geodesic path.</td>
</tr>
<tr>
<td>(\exp_p(v))</td>
<td>The exponential of (v) at base point (p)</td>
</tr>
<tr>
<td>(\log_p(x))</td>
<td>The logarithmic of (x) at base point (p)</td>
</tr>
<tr>
<td>(d(x, y))</td>
<td>The geodesic distance between (x) and (y)</td>
</tr>
<tr>
<td>(\mathbb{R})</td>
<td>The set of all real numbers</td>
</tr>
<tr>
<td>(\mathbb{H})</td>
<td>The set of all quaternions</td>
</tr>
<tr>
<td>(S^1)</td>
<td>The 1-sphere, i.e. the circle in a 2-dimensional space</td>
</tr>
<tr>
<td>(S^2)</td>
<td>The 2-sphere, i.e. the sphere in a 3-dimensional space</td>
</tr>
<tr>
<td>(S^n)</td>
<td>The (n)-sphere, i.e. a sphere in a ((n + 1))-dimensional space</td>
</tr>
<tr>
<td>(\mathbb{R}P^2)</td>
<td>The real projective plane</td>
</tr>
<tr>
<td>(\mathbb{R}P^3)</td>
<td>The real projective space</td>
</tr>
<tr>
<td>(\mathbb{R}P^n)</td>
<td>The real projective (n)-space</td>
</tr>
<tr>
<td>(SO(3), SO(3, \mathbb{R}))</td>
<td>The (real) special orthogonal group in 3 dimensions</td>
</tr>
</tbody>
</table>
The theory of smooth Riemannian manifolds is well known in mathematics, but perhaps less known to practitioners of machine learning, signal- and image processing. In the following we first review some useful linear algebra, introduce tensors, geometric entities defined in a vector space, and finally Riemannian manifolds that generalize the concept of curved surfaces. The intention is to give brief overview and provide some references to further reading.

2.1 Linear algebra

To be able to introduce tensors, it is convenient to first define vectors, vector spaces and related concepts.

2.1.1 Vector spaces

Let $V$ be a vector space with $\dim(V) = n$. A basis for $V$ is a set of elements $B = \{b_1, b_2, \ldots, b_n\} \subset V$ which are linearly independent and spans $V$, i.e. for any vector $v \in V$ there is a set of coordinates $x^i$ such that

$$v = \sum_{i=1}^{n} x^i b_i$$  \hspace{1cm} (2.1)

and

$$\sum_{i=1}^{n} x^i b_i = 0$$  \hspace{1cm} (2.2)

has the unique solution $x^i = 0$. 
2.1.2 Linear maps

A linear map $f$ is a map between two vector spaces $V$ and $W$, $f : V \rightarrow W$, that is additive and homogeneous:

$$f(u + v) = f(u) + f(v) \quad \text{(2.3)}$$

$$f(\lambda u) = \lambda f(u) \quad \text{(2.4)}$$

2.1.3 The dual vector space

The dual vector space $V^*$ is the space of all linear maps $w : V \rightarrow \mathbb{R}$. Thus $w(u) \in \mathbb{R}$ and

$$w(u + v) = w(u) + w(v) \quad \text{(2.5)}$$

$$w(\lambda u) = \lambda w(u) \quad \text{(2.6)}$$

A simple example is the function $w(v) = a \cdot v$, where $u, v \in \mathbb{R}^n$. Or more general, $w(v) = \langle a, v \rangle$ where $u, v \in V$, for some $V$ equipped with an inner product. $V^*$ is a vector space with dim$(V^*) = n$. An element $w \in V^*$ operating on a vector $v = \sum_{i=1}^{n} x_i b_i$ may be decomposed,

$$w(v) = w\left(\sum_{i=1}^{n} x_i b_i\right) = \sum_{i=1}^{n} x_i w(b_i) = \sum_{i=1}^{n} x_i w_i. \quad \text{(2.7)}$$

Apparently, the action on the elements of a basis $B$ uniquely determines the action on any vector expressed in that basis.

A dual basis to $B$, $W = \{b^i\}$, is defined by

$$b^i(b_j) = \delta^i_j \quad \text{(2.8)}$$

where $\delta^i_j$ is the Dirac delta function,

$$\delta^i_j = \begin{cases} 
1 & \text{if } i = j \\
0 & \text{otherwise} 
\end{cases} \quad \text{(2.9)}$$

$V$ and $V^*$ are different vector spaces and there is not necessarily a way to identify a vector $v \in V$ with an element $w \in V^*$, unless there is an inner product defined. Then $v$ may be identified with the element $w$, defined by $w(u) = \langle v, u \rangle$. One interpretation of a dual vector is that it measures some aspect of an ordinary vector. If ordinary vectors are geometrically depicted as arrows of different length, a dual vector can be thought of as the slope of a scalar function defined in $V$ or a level curve to a linear scalar function in $V$, see Fig. 2.1.

From Eq. 2.8 we note the convention that ordinary or contravariant basis vectors are written in boldface with a lower index, $b_i$, while covariant basis vectors are written using boldface with an upper index, $b^i$. Consequently, the coordinates of
2.1 Linear algebra

Figure 2.1: (a): a contravariant vector $x^i$. (b): A contravariant vector $2x^i$. (c): A covariant vector $w_i$ and various contravariant vectors $z^i$, for which $z^i w_i = 1$. (d): A covariant vector $2w_i$. Note that the graphical representation or glyph, of a contravariant vector, which can be thought of as a level curve of a scalar function, gets narrower when the coefficients are doubled. This behavior is different from the arrow representing a contravariant vector, which gets longer when the coefficients are doubled.

A covariant vector are denoted $x^i$, and the coordinates of a covariant vector are with a lower index, $w_i$. From now on, a vector is often denoted by its coordinates, $x^i$ or $w_i$, which is practical since it then becomes possible to distinguish between contravariant and covariant vectors. Sometimes we also use the notation $v$, usually to denote a contravariant vector, when there is no way to mix up covariant and contravariant vectors. This notation is the most well-known notation for most readers after all.

2.1.4 The Einstein summation convention

Since many expressions involving vectors, matrices and soon also tensors, include summations, it is now time to introduce the so-called Einstein summation convention. It means that indices that occur in several places in an expression are summed over from 1 to $n$, where $n = \dim(V)$, e.g.

$$v = \sum_{i=1}^{n} x^i b_i = x^i b_i$$

(2.10)

or

$$w(v) = w(\sum_{i=1}^{n} x^i b_i) = \sum_{i=1}^{n} x^i w_i b^i(b_i) = x^i w_i.$$  

(2.11)

For vectors, this results in a slightly shorter notation. For higher order tensors however, this notation is more even practical.

2.1.5 Coordinate changes

Coordinate changes in the vector space $V$ induce a dual coordinate change in $V^*$ if the dual basis is assumed. Let $x^i$ denote $\sum_{i=1}^{n} x^i b_i$ and $w_i$ denote $\sum_{i=1}^{n} w_i b^i$. 

Introduce a coordinate change in the contravariant coordinates, \( \tilde{x}^i = t^i_j x^j \). Then regardless of coordinate system, we have

\[
\begin{align*}
\tilde{x}^i w_i &= \tilde{x}^i \tilde{w}_i \\
&= x^j t^i_j T^k_i w_k \Rightarrow \\
t^i_j T^k_i &= \delta^k_j
\end{align*}
\] (2.12) (2.13) (2.14)

for some coordinate change \( T^k_i \) in the dual space. Thus, coordinates of dual vectors in \( V^* \) must transform inversely to coordinate changes in \( V \). The following example 2.1.1 gives some intuition to coordinate changes from a simple example in physics.

**Example 2.1.1.** Consider a capacitance consisting of two charged metal plates separated by a gap \( d = 0.5 \text{m} \) with a potential difference \( U = 100 \text{V} \), depicted in Fig. 2.2. Then the field strength \( E = 200 \text{V/m} \), since it satisfies the equation \( U = d \cdot E \). By changing the spatial coordinate system from meter to feet we obtain \( d = 1.64 \text{ft} \), \( U = 100 \text{V} \) and \( E = 60.98 \text{V/ft} \). Length is a contravariant vector and the coordinate of \( d \) increases during the coordinate change. Fields strength is a gradient, a covariant vector, and is coordinate decreases from this coordinate change. Thus, there are two types of vectors, covariant and contravariant, which are dual. The type of a vector is often hinted by the associated physical unit, i.e. whether the spatial unit (m, ft, . . .) is in the numerator or denominator, as seen in the example above.

**2.1.6 Inner products and metrics**

An inner product \( \langle \mathbf{u}, \mathbf{v} \rangle \), or equivalently for our purposes a metric \( g(\mathbf{u}, \mathbf{v}) \), is a bi-linear map (linear in each argument) \( g : V \times V \rightarrow \mathbb{R} \) with two additional
2.2 Tensors

Tensors generalize scalars, vectors and matrices to higher dimensions. Sometimes the word “tensor” is used for any multi-dimensional array with more indices than a matrix, more than two, but we use the term in a more precise meaning that is in agreement with the notation in physics and differential geometry. In these research fields tensors are geometric objects that are invariant under coordinate changes, just like vectors. In physics the word “tensor” usually refers to what in mathematics would be called a “tensor field” but in both domains it is meaningful to think of tensors as objects defined pointwise in a vector space.

Many spatial quantities in physics are tensors, for instance: velocity (m/s), diffusion (m²/s) and electric field strength (V/m). In mathematics, contravariant vectors are those that behave like we are used to, while the covariant vectors are gradients. Examples of higher order tensors in mathematics are quadratic forms.

A tensor $F$ is defined as multi-linear map,

$$ F : V^* \times \ldots \times V^* \times V \times \ldots \times V \to \mathbb{R}, $$ (2.22)

i.e. a map that is linear in each of its arguments. Its order is $r + s$ and it has type $(r, s)$, meaning that it operates on $r$ covariant tensors and $s$ contravariant tensors.

In some contexts, order is called rank and type is called valence, which can be
confusing since rank is also used to describe the rank of matrices. Similar to vectors and the metric previously defined, the action of tensors can be defined by components that are derived from the action on all combinations of basis vectors \( \{ w^i \} \) in \( V^* \) and \( \{ b_j \} \) in \( V \),

\[
F_{i_1,i_2,\ldots,i_r}^{j_1,j_2,\ldots,j_s} = T(w^{i_1}, \ldots, w^{i_r}, b^{j_1}, \ldots, b^{j_s}).
\] (2.23)

The number of components is \( n^{r+s} \). If the coordinates are changed, \( \tilde{x}^i = t_k^ix^k \), then each contravariant index is transformed as a vector and each covariant index is transformed as a dual vector,

\[
\tilde{F}_{x^yz}^{abc} = F_{x^yz}^{abc} t_a^i t_b^j t_c^k \ldots (t^{-1})^x_m(t^{-1})^y_n(t^{-1})^z_o \ldots
\] (2.24)

In physics, this is sometimes how tensors are defined, i.e. as objects that transform according to certain transformation laws.

### 2.2.1 Outer products

The outer product of two vectors, \( F \) and \( G \), having type \((r, s)\) and \((p, q)\), is defined by

\[
(F \otimes G)((x_1)_a, \ldots, (x_{r+p})_a, (y_1)_a, \ldots, (y_{s+q})_a)
\]

\[
F((x_1)_a, \ldots, (x_r)_a, (y_1)_a, \ldots, (y_s)_a) G((x_1)_a, \ldots, (x_p)_a, (y_1)_a, \ldots, (y_q)_a)
\]

where \((x_i)_a\) refers to the i:th covariant vector.

### 2.2.2 Cartesian tensors

It is common in e.g. continuum mechanics to work solely using Cartesian vectors and tensors. This means that an ON basis is used and the basis and dual basis coincide and there is no need to differentiate between upper and lower indices.

### 2.2.3 Index gymnastics

Many operations in tensor analysis can be performed by manipulation of the indices, which is sometimes known as index gymnastics. A contravariant vector \( x^i \) may for instance be transformed to a covariant vector by multiplication with the metric \( g_{ij}, \ x_i = g_{ij}x^j \). It is called to “lower” an index. In a similar fashion, an index may be “raised”, \( w^i = g^{ij}w_j = (g^{-1})^{ij}w_j \).
2.3 Manifolds

Manifolds generalize curves and surfaces to higher dimensions and generalize Euclidean geometry to arbitrary curved spaces. The general term “manifold”, which we will use frequently, actually refers to a structure in which every point has a neighborhood that looks like the Euclidean space. We will most frequently mean a Riemannian manifold, which is a differentiable manifold equipped with a metric, i.e. an inner product in each of its tangent spaces. One may visually think of a manifold as a surface embedded in $\mathbb{R}^3$ but the theory of manifolds is possible to introduce without the need for an embedding space. In general relativity for instance, it is known that the 4-D space-time is curved, but there is no need for a higher dimensional space in which it is embedded.

2.3.1 Charts and atlases

A manifold is defined by a set of open subsets $U_i \subset \mathbb{R}^n$, to which maps are defined from the manifold $M$, see Fig. 2.3. These open subsets overlap and through the use of inverse mappings, via the manifold $M$, it is possible to walk from one $U_i$ to another. These subsets and maps are generally known as charts, or coordinate systems in physics. Many such charts may be collected to form an atlas of a manifold. In this dissertation, we will frequently use charts of a manifold, but we will only deal with problems where one chart is enough.

![Figure 2.3](image)

**Figure 2.3**: An illustration how different $\Phi_i$ maps subsets of the manifold $M \subset \mathbb{R}^3$ to various charts in $\mathbb{R}^2$

2.3.2 The tangent space

In each point on the manifold, there is a tangent space $T_pM$ defined, consisting of the directional derivatives along curves passing through this particular point. $T_pM$ is thus spanned by basis vectors $\frac{\partial}{\partial x^i}$. In every tangent space there is a metric
defined, generally denoted $g_{ij}$. This allows for the calculation of e.g. lengths, angles and area inside the manifold.

### 2.3.3 Geodesic length and distance

The metric (inner product) $g_{ij}$ defined in the tangent space to a point $p$, $T_p M$, of a manifold $M$, allow the measurement of lengths of tangent vectors. I.e. if $x^i \in T_p M$, $||x^i|| = \sqrt{g_{ij}x^jx^i}$. This allows for the definition of the length of a curve $c : [a, b] \rightarrow M$ by

$$\int_a^b ||\dot{c}(t)|| dt.$$  \hspace{1cm} (2.25)

The geodesic distance between two points $p$ and $q$ is defined by the minimum length over all curves connecting $p$ and $q$, i.e. $c(a) = p$ and $c(b) = q$.

$$d(p, q) = \min_{c: c(a) = p, c(b) = q} \int_a^b ||\dot{c}(t)|| dt.$$  \hspace{1cm} (2.26)

### 2.3.4 Further reading

For a more complete introduction to manifolds, we refer the reader to the individual chapters in this dissertation. These chapters are more or less self-contained and despite the topic of this dissertation, we only use the most basic concepts from manifolds and differential geometry. The notion of comma derivative (partial derivative), semicolon derivative (covariant derivative) and the $\exp$ and $\log$ maps are defined when they are needed. The notation of vectors and tensors, including the Einstein summation convention, is probably what will confuse a reader who is unfamiliar with the topic. We also refer to introductory books on differential geometry, for instance (Wald, 1984), (Isham, 1989) and (do Carmo, 1992).
3

Dimension reduction and manifold learning

3.1 Machine learning

Visualization, processing and analysis of high-dimensional data such as images often requires some kind of pre-processing to reduce the dimensionality of the data and find a mapping from the original representation to a low-dimensional vector space. The assumption is that the original data resides in a low-dimensional subspace or manifold, embedded in the original space. This topic of research is called dimensionality reduction, non-linear dimensionality reduction or more recently manifold learning.

The class of methods for dimension reduction and manifold learning is quite broad and the criteria for finding a low-dimensional parameterization varies. One of the most well-known algorithms is PCA, Principal Components Analysis, which projects data onto the $n$-dimensional linear subspace that maximizes the variance of the data in the new space.

If the original data points lie on a manifold $M$, the mapping to a new space $N$ may give an embedding or an immersion of the original manifold. In differential geometry, an immersion corresponds to a smooth mapping $f(x)$ for which the differential of $f(x)$, $d_xf(x) : T_pM \rightarrow T_{f(p)}M$, is non-singular and injective. When the mapping $f(x)$ itself is also injective, it corresponds to an embedding. An example of an embedding is the mapping of a set of pictures (high-dimensional) of a clock to a representation on the unit circle in $\mathbb{R}^2$. An immersion could then be a mapping to a curve in $\mathbb{R}^2$ shaped like the figure “8”. Also, see Fig. 3.1 for some intuitive examples.

3.1.1 Dimensionality reduction

The use of linear methods for dimensionality reduction is a rather mature area of research, starting with PCA, Principal Components Analysis (Pearson, 1901) a.k.a. the Hotelling transform (Hotelling, 1933) and the Karhunen-Loève Trans-
18 Chapter 3. Dimension reduction and manifold learning

Figure 3.1: Top-Left: A 1-D manifold embedded in $\mathbb{R}^2$. Top-Right: A 1-D manifold immersed in $\mathbb{R}^2$. Bottom-Left: The torus, a 2-D manifold embedded in $\mathbb{R}^3$. Bottom-Right: Boy’s surface, an immersion of the projective plane $\mathbb{R}P^2$ in $\mathbb{R}^3$.

form (Karhunen, 1947). Variants of PCA include generalizations such as Empirical Orthogonal Functions (Lorentz, 1956) and Kernel Principal Components Analysis (Schölkopf et al., 1998). See figure 3.2 for a schematic view of linear methods for dimension reduction.

The basic idea in PCA is to find a projection of the data that maximizes variance. For a set of vectors $x_i \in \mathbb{R}^N$, this can be done by the following procedure.

1. Calculate the $N \times 1$ sample mean vector, $\bar{u} = \frac{1}{M} \sum_{i=1}^{M} x_i$.
2. Subtract mean from the data points $\tilde{x}_i = x_i - \bar{u}$
3. Organize $\tilde{x}_i$ into a $N \times M$ matrix $X$.
4. Create the sample covariance matrix $C = \frac{1}{M-1} \tilde{X} \tilde{X}^T$.
5. Calculate the $K$ largest eigenvalues of $C$ and store the corresponding eigenvectors in a $N \times K$ matrix called $W$.
6. Projections on the PCA basis may now be calculated as $y_i = W^T(x_i - \bar{u})$.

PCA has been widely used; “eigenfaces” (Turk and Pentland, 1991) is one of the more well-known applications where it is used to create a low-dimensional linear subspace describing variations in images of human faces. The Karhunen-Loève transform is also known to be useful to create natural basis functions for image
3.1 Machine learning

Another well-known linear method to find embeddings or immersions of data points, possibly sampled from a manifold, is Multidimensional Scaling (MDS) (Torgerson, 1952; Young and Householder, 1938). Instead of preserving variance in the projection, it strives to preserve all pairwise distances during the projection. Similar to PCA, the basic variant of Multidimensional Scaling is possible to calculate by solving an eigenvalue problem. This is attractive since eigenvalue problems are optimization problems for which efficient and globally convergent algorithms exist. The classic MDS is stated as a minimization problem of finding new low-dimensional coordinates \( y_i \) for the dataset \( x_i \) given all pairwise Euclidean distances \( d(x_i, x_j) \). The solution, up to a rotation, is given by

\[
\{y_i\} = \arg\min_{\{y_i\}} \sum_{i,j=1}^{M} (d(x_i, x_j)^2 - \|y_i - y_j\|^2)^2
\]  (3.1)

Important to note is that classical MDS works with quadratic distances, which might seem unnatural but makes it possible to solve the minimization problem by the solution of an eigenvalue problem. If distances correspond to Euclidean distances, classical MDS is equivalent to PCA.

Variants of MDS include non-metric Multidimensional Scaling and weighted MDS. In weighted MDS the objective function is replaced,

\[
\{y_i\} = \arg\min_{\{y_i\}} \sum_{i,j=1}^{M} w_{ij} (d(x_i, x_j) - \|y_i - y_j\|)^2.
\]  (3.2)

This objective function differs from classical MDS. It does not fit squared distances. As a consequence, this objective function might have several local minima and eigen-decomposition cannot be used to solve the problem in one step. Therefore, some strategy for coping with local minima should be employed in the numerical minimization procedure. The benefit of weighted MDS is that uncertainty and missing data can be modeled using appropriate weights.
Other important linear projections of data in vector spaces include Projection Pursuit (Friedman and Tukey, 1974) and Independent Component Analysis (Jutten and Herault, 1991). A well-known related example for non-metric data is Latent Semantic Indexing or LSI (Berry et al., 1995). LSI maps document-vectors, describing the occurrences of words in documents, to a low-dimensional vector space.

### 3.1.2 Manifold learning

Recently there has been a great interest in methods for parameterization of data using low-dimensional manifolds as models. Within the neural information processing community, this has become known as manifold learning. Methods for manifold learning are able to find non-linear manifold parameterizations of data-points residing in high-dimensional spaces, very much like Principal Component Analysis (PCA) is able to learn or identify the most important linear subspace of a set of data points. In two often cited articles in Science, Roweis and Saul introduced the concept of Locally Linear Embedding (Roweis and Saul, 2000) and Tenenbaum et al. introduced the so-called Isomap (Tenenbaum et al., 2000). This seems to have been the start of the most recent wave of interest in manifold learning.

![Figure 3.3: A schematic view of the fitting of 1-D non-linear manifold to a set of data points embedded in 2-D.](image)

Early work was done by Kohonen with the so-called Self-Organizing Maps (SOM) (Kohonen, 1982), in which a grid of points that is fitted to the data set provides a topologically constrained model of a manifold. This work was later improved in the Generative Topographic Map (GTM) (Bishop et al., 1998). Bregler and Omohundro were also early in adopting the view of data as points on a non-linear manifold in a vector space, modeling the manifold of lip images (Bregler and Omohundro, 1994). A non-linear variant of PCA, called Kernel Principal Components Analysis (KPCA) (Schölkopf et al., 1998), has also been introduced. In KPCA, the input vectors are mapped to a new feature space before applying PCA, a procedure that is performed implicitly through the notion of an inner product or kernel. Later, contemporary with Isomap and LLE, Belkin and Niyogi described
how approximations to the Laplacian operator and heat equation can be used to perform manifold learning in their framework called Laplacian Eigenmaps (LE) (Belkin and Niyogi, 2002).

### 3.1.3 Laplacian eigenmaps

As an example of a method for manifold learning, we first mention Laplacian Eigenmaps (Belkin and Niyogi, 2002). The basic algorithm consists of three steps:

1. First a graph is constructed where each node corresponds to a data point $x_i$. Edges are created to each of the $K$ nearest neighbors of $x_i$. See figure 3.4.

2. Weights are then assigned to each edge in the graph, for instance using a Gaussian kernel to give strong weight to edges connecting data points that are close in the original space. The weights are collected in a matrix $W_{ij}$.

3. To find a low-dimensional embedding $\{y_i\}$ corresponding to $\{x_i\}$, define an objective function $V$ that has a low value when nodes with a strong edge are mapped close to each other.

$$V(\{y_i\}) = \frac{1}{2} \sum_{i,j} ||y_i - y_j||^2 W_{ij} \quad (3.3)$$

Define a diagonal matrix $D$, such that $D_{ii} = \sum_j W_{ij}$ and the Laplacian matrix $L = D - W$. If $Y$ gives the $m$-dimensional coordinates of $y_i$ on the $i$th row of $Y$, and the constraint $Y^TDY = I$ is added, the Laplacian eigenmap of dimension $m$ is now found by the solution of the eigenvalue problem $Lv = \lambda Dv$. If the eigenvectors $\{v^{(0)}, v^{(1)}, \ldots, v^{(N-1)}\}$ are ordered after the size of the eigenvalues, the first being the smallest (actually equal to 0), then $\hat{Y} = (v^{(1)}, v^{(2)}, \ldots, v^{(m)})$ gives the solution for the optimal embedding, minimizing the value of $V$.

Figure 3.4: A schematic view of the formation of a graph by connecting nearby samples.

The Laplacian Eigenmaps is sometimes referred to as a local method for manifold learning, meaning that it is an attempt to preserve local geometrical properties in the mapping to a low-dimensional space (de Silva and Tenenbaum, 2002).
3.1.4 Isomap – isometric feature mapping

An example of a global method for manifold learning is Isomap (Tenenbaum et al., 2000). It tries to preserve the geometry of the data manifold in all scales, mapping nearby points to nearby points and faraway points to faraway points (de Silva and Tenenbaum, 2002). The basic steps of the algorithm are:

1. Create a neighborhood graph $G$ for the dataset $\{x_i\}$, based for instance on the $K$ nearest neighbors of each point $x_i$.
2. For every pair of nodes in the graph, compute the shortest path as an estimate of intrinsic distance within the data manifold. The edges of the graph are weighted according to the Euclidean distance between the corresponding data points.
3. Use the intrinsic distance estimates as input to classical MDS and find an optimal $m$-dimensional embedding $\{y_i\}$.

The convergence properties of the estimation procedure for the intrinsic distances are further described in (Bernstein et al., 2000).

Computing $N \times N$ pairwise distances is a computationally heavy operation, and so is solving a large eigenvalue problem. In comparison to for instance Laplacian Eigenmaps, the eigenvalue problem in Isomap is not sparse. A variation of Isomap is the L-Isomap, based on the so-called Landmark MDS method. It works by first calculating the Isomap embedding for $n$ points, the landmarks, selected at random. Then the solutions for the rest of the points are computed by an interpolation technique similar to triangulation. This technique is also very similar to the proposed method for calculating the sample LogMap, and even though the two approaches are different in philosophy, they share some obvious similarities.

The interpolation procedure is the following for a point $x_i$ that is not a landmark. Let the $m$-dimensional landmark coordinates be column vectors in a $m \times n$ matrix $L$. Let $\Delta_n$ be the squared distance matrix for all pairs of landmarks and $\overline{\Delta}_n$ the column mean of $\Delta_n$. Let $\Delta_i$ be a column vector of all squared distances from $x_i$ to the landmarks. Also, assume that the landmarks are centered. Then the interpolated coordinate is given by

$$y_i = \frac{1}{2} L^\dagger (\overline{\Delta}_n - \Delta_i)$$

(3.4)

where $\dagger$ denotes the Moore-Penrose pseudoinverse. This is basically an estimate of $-1/2$ times the derivative of the squared distance function to $x_i$, evaluated at the origin.

3.1.5 A brief historical timeline

A full review of dimension reduction and manifold learning is out of scope for this thesis. The activity in this field is increasing and the following list is a brief
3.1 Machine learning

summary, which may also serve as a timeline.

- Principal Components Analysis, PCA (Pearson, 1901; Hotelling, 1933) or (Karhunen, 1947).
- Multidimensional Scaling, MDS (Young and Householder, 1938; Torgerson, 1952)
- Empirical Orthogonal Functions, EOF (Lorentz, 1956)
- Projection Pursuit, PP (Friedman and Tukey, 1974)
- Self Organizing Maps, SOM (Kohonen, 1982)
- Principal Curves (Hastie and Stuetzle, 1989)
- Surface Learning with Applications to Lip Reading (Bregler and Omohundro, 1994)
- Curvilinear Component Analysis, CCA (Demartines and Herault, 1997)
- Generative Topographic Mapping (Bishop et al., 1998)
- Kernel Principal Components Analysis, KPCA (Schölkopf et al., 1998)
- Isometric feature mapping, Isomap (Tenenbaum et al., 2000) and C-Isomap and L-Isomap (de Silva and Tenenbaum, 2002).
- Locally Linear Embedding, LLE (Roweis and Saul, 2000)
- Laplacian Eigenmaps, LE (Belkin and Niyogi, 2002)
- Local Tangent Space Alignment, LTSA (Zhang and Zha, 2002)
- Hessian Eigenmaps, HLLE (Donoho and Grimes, 2003)
- Relational Perspective Map, RPM (Li, 2004)
- Semidefinite embedding (Weinberger and Saul, 2004)
- Diffusion Maps (Nadler et al., 2006)
- Non-Isometric Manifold Learning (Dollár et al., 2007)

In general, linear methods for dimension reduction are more stable and more mature. Principal Components Analysis and Multidimensional Scaling are still very popular and have the advantage of being able to learn meaningful relations from few samples. Some of the oldest methods for manifold learning, such as the Self Organizing Feature Maps, have also been used in many applications and may be considered as mature from an application point of view. The more recent methods for manifold learning have mainly two advantages: 1) they are based on global optimization and the solution of eigenvalue problems or semi-definite programming
(unlike SOMs which are sensitive to local minima in the objective function). 2) they have shown to be efficient for datasets where linear methods fail, such as the simple “Swiss roll” dataset (Tenenbaum et al., 2000; Roweis and Saul, 2000).
4

Diffusion tensor MRI

4.1 Diffusion imaging

In the physical world, diffusion is the collective process of random motion of particles in a solution or gas. On a macroscopic scale, this phenomenon is visible to the eye, for instance by adding a drop of ink to a glass of water and watching it dissolve. The process, also known as Brownian motion, was named after the Scottish botanist Robert Brown who observed the random motion of individual plant spores in a water solution using a microscope. In 1905, Albert Einstein presented a theoretical analysis of Brownian motion and linked it to the Boltzmann constant.

Today diffusion processes are fundamental for the understanding of both physics and mathematics. In Magnetic Resonance Imaging, MRI, it is possible to measure and visualize the diffusion of water molecules inside living organisms. This technology, called Diffusion-Weighted MRI, is today a part of clinical practice, e.g. for the diagnosis of stroke. More recent methods, such as Diffusion Tensor MRI combined with so-called fiber tractography, are able to in vivo infer the anatomy and connectivity of nerve bundles within white matter in the human brain. The usefulness of this, for morphological or functional studies of the brain, or to perform surgical planning before the removal of a tumor, is evident.

DT-MRI is perhaps also the canonical example of how important the mathematical modeling using Riemannian geometry is for medical imaging today. The estimated tensor field in DT-MRI may be interpreted as a metric, see for instance (O’Donnell et al., 2002), and the space of diffusion tensors has been modeled using Riemannian geometry to perform accurate interpolation and filtering of diffusion tensor fields (Batchelor et al., 2005; Pennec et al., 2005; Fletcher and Joshi, 2007; Arsigny et al., 2006a; Kindlmann et al., 2007).
4.1.1 Diffusion

To get some intuition on diffusion processes, consider the following example of coin flipping. Let two players, player A and player B, flip a coin. If heads come up, player B gives one dollar to player A. If tails come up, A gives one dollar to B. Call the profit for player A after $n$ turns $a(n) \in [-n, n]$ and let $a(0) = 0$. Each turn of the game, $a(n + 1)$ is either $a(n) + 1$ or $a(n) - 1$, and the variable $a(n)$ perform a random walk in $\mathbb{Z}$. Whether A or B is the winner after $n$ turns in a particular game is impossible to say from the beginning, but the variance, $\text{Var}(a(n)) = E\{a(n)^2\}$, after many games lasting for $n$ turns, is easy to calculate. The total variance of the sum of $n$ independent variables, each with a variance 1, is $n$. Thus $\text{Var}(a(n)) = n$, meaning that the variance of the profit is growing linearly with the respect to the number of turns in the game.

The diffusion coefficient

Translating the example of coin flipping to particles performing a random walk in discrete time in one dimension, the variance is growing linearly if the jumps of the particle are according to a set of independent and identically distributed (i.i.d.) variables. Generalizing to continuous time, a natural physical unit to measure the strength of diffusion is $m^2/s$.

Diffusion in a 3-D isotropic medium is in a similar way characterized by the diffusion coefficient, $c$. The variance of the distance, $|r|$, a particle moves by a random walk during time $t$ is $\text{Var}(|r|) = 6ct$. Looking at the individual dimensions, we have $\text{Var}(r_x) = \text{Var}(r_y) = \text{Var}(r_z) = 2ct$.

The diffusion tensor is a generalization of $c$ to account for anisotropic diffusion in three dimensions. It is defined as $D = \frac{\text{Var}(r)}{2t} = \frac{E\{rr^T\}}{2t}$. Similar to the variance, it is a second order contravariant tensor, described by a symmetric positive semidefinite $3 \times 3$-matrix. Using $D$, we may measure the diffusion coefficient along a particular direction $\hat{g}$ by the formula $c(\hat{g}) = \hat{g}^T D \hat{g}$. In an isotropic medium the diffusion tensor (in a ON basis) simply becomes

$$D = \begin{pmatrix} c & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & c \end{pmatrix} \quad (4.1)$$

The apparent diffusion coefficient

The diffusion coefficient and the diffusion tensor both describe the behavior of unrestricted diffusion. For water molecules in biological tissue, the diffusion is
often restricted by for instance cell membranes. For short time intervals, the diffusion of a single molecule is governed by the diffusion tensor or the diffusion coefficient. On a larger time scale however, collisions with boundaries of various kinds will restrict diffusion. This will affect the measurement of diffusion and the term apparent diffusion coefficient (ADC) is used instead.

### 4.1.2 Estimating diffusion tensors

Using diffusion-weighted MRI, it is possible to measure the apparent diffusion coefficient in different directions. The Stejskal-Tanner equation relates measurements to ADC values:

\[ S_k = S_0 e^{-\gamma^2 \delta^2 [\Delta - (\delta/3)]} \]

(4.2)

A generalization to diffusion tensors \( \mathbf{D} \) and gradient directions \( \mathbf{g} \) is straightforward.

\[ S_k = S_0 e^{-\gamma^2 \delta^2 [\Delta - (\delta/3)] \mathbf{g}^T \mathbf{D} \mathbf{g}} \]

(4.3)

In the equation above, \( \gamma \) is the proton gyromagnetic ratio (43MHz / Tesla), and \( \mathbf{g} \) is the gradient field vector, \( \delta \) is the duration of the diffusion gradient pulses and \( \Delta \) is the time between the diffusion gradient RF pulses. The value \( S_k \) refers to the measured signal, attenuated by diffusion, and \( S_0 \) is the corresponding value obtained when the diffusion gradient strength is zero.

Estimation of \( \mathbf{D} \) from a series of diffusion-weighted measurements is possible, either using a least squares approach (Westin et al., 2002) or using statistical methods. The unknown values are \( S_0 \) and \( \mathbf{D} \), containing in total 7 degrees of freedom (due to the symmetry of \( \mathbf{D} \)). See figure 4.1 for a set of 8 images used in DT-MRI (two of the images are averaged before estimation begins). The measurements \( S_k \) will be affected by Rician distributed noise (Gudbjartsson and Patz, 1995) from the MRI acquisition process.

### 4.1.3 Diffusion in the human brain

Inside the human brain, the apparent diffusion properties will vary depending of the type of tissue. In table 5.1 the ADC has been measured for various tissues. The different eigenvalues mentioned will be explained in more detail below, but refers to the fact that diffusion varies in different directions – the diffusion tensor \( \mathbf{D} \) is anisotropic – for certain types of tissue, in particular inside white matter (WM).

Close to fiber structures in the brain, the diffusion of water molecules is restricted. The variance of the random walk is attenuated in directions perpendicular to the fibers, while the movement along the fibers is similar to free diffusion. The anisotropy of the apparent diffusion is captured in the diffusion tensor. By studying the main direction of diffusion, derived from the eigenvalues and eigenvectors...
Figure 4.1: A total of eight axial slices of a human brain have been acquired to calculate one slice of diffusion tensors. The six first images are diffusion-weighted and have been collected with non-zero gradients in six different gradient directions $\hat{g}$. The two last images have been collected with zero gradients, $g = 0$.

<table>
<thead>
<tr>
<th>Eigenvalues of $D$ ($10^{-6}\text{mm}^2/\text{s}$)</th>
<th>Pyramidal tract (WM)</th>
<th>Splenium of the corpus callosum (WM)</th>
<th>Optic radiation (WM)</th>
<th>Caudate Nucleus (GM)</th>
<th>Cerebrospinal fluid (CSF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_1$</td>
<td>$1,708 \pm 131$</td>
<td>$1,685 \pm 121$</td>
<td>$1,460 \pm 75$</td>
<td>$783 \pm 55$</td>
<td>$3,600 \pm 235$</td>
</tr>
<tr>
<td>$\lambda_2$</td>
<td>$303 \pm 71$</td>
<td>$287 \pm 71$</td>
<td>$496 \pm 59$</td>
<td>$655 \pm 28$</td>
<td>$3,131 \pm 144$</td>
</tr>
<tr>
<td>$\lambda_3$</td>
<td>$114 \pm 12$</td>
<td>$109 \pm 26$</td>
<td>$213 \pm 67$</td>
<td>$558 \pm 17$</td>
<td>$2,932 \pm 212$</td>
</tr>
</tbody>
</table>

Table 4.1: Typical ADC values found in the human brain, measured in the orientations of the three eigenvectors of $D$ (Pierpaoli et al., 1996).

of the diffusion tensor, it is possible to infer the orientation of fibers going through a voxel. This forms the basis for fiber tracking. Studying the degree of anisotropy of a diffusion tensor also give a lot of information about the organization of tissue within that specific voxel.

4.1.4 Applications of DT-MRI

The applications of DT-MRI in a clinical setting include examples of both quantitative and qualitative methods.
4.2 Processing diffusion tensor data

Surgical planning

During surgical planning involving the brain, knowledge of the location of important fiber bundles may guide the surgeon to avoid damage on important functional parts of the brain. This is particularly important when planning the removal of tumors, while fiber tracts may have been distorted by the growth of the tumor so that experience and prior knowledge of fiber bundles are of little importance in the case at hand.

Clinical studies

For morphological and functional studies of the human brain, in both healthy populations and patients, diffusion tensor MRI can be useful to derive knowledge related to white matter variations and abnormalities. This includes for instance studies on Schizophrenia and Multiple Sclerosis. With DT-MRI it is also possible to perform non-invasive and harmless experiments on human subjects to find out about pathways in the brain, and confirm hypotheses about the human brain derived from invasive and dangerous studies previously only performed on animals and in particular monkeys.

4.2 Processing diffusion tensor data

Processing and analysis of tensor-valued data in image volumes requires a treatment different from that of scalar data. While image processing for tensor images was available prior to the introduction of DT-MRI, see for instance (Knutsson, 1989; Granlund and Knutsson, 1995), the recent advances in acquisition of tensor-valued data in medicine (Westin et al., 2002) has made this field of research popular again.

4.2.1 Scalar invariants

Tensors and tensor volumes are more difficult to visualize and analyze than scalars and scalar-valued volumes. For this reason, methods for calculating scalar values derived from tensors are important; in particular methods that yields scalars that are invariant to rotations of the coordinate frame. Three important invariants are the trace, fractional anisotropy and the shape classification of tensors by Westin.

Trace

The trace of the tensor is defined

$$Tr(D) = \sum_{i=1}^{n} D_{ii}$$  \hspace{1cm} (4.4)
For a mixed second order tensor, the trace is a scalar that is invariant to changes of basis and thereby invariant to rotations. While the diffusion tensor is a contravariant tensor, $D^{ij}$, and the trace is only defined for mixed tensors, it is necessary to first transform the diffusion tensor $D^{ij}$ to a mixed tensor $D_{ij} = D^{ik}g_{kj}$. Using the trace, a mean diffusion coefficient can be calculated using

$$\bar{c} = \frac{1}{3} Tr(D^{ij}) = \frac{1}{3} Tr(D^{ik}g_{jk}) = \frac{1}{3} D^{ik}g_{ik} = \frac{1}{3} \sum_{i=1}^{n} \sum_{k=1}^{n} D^{ik}g_{ik} \quad (4.5)$$

This scalar invariant is formed by letting the metric tensor operate on the diffusion tensor. It is thus dependent of the choice of unit used to define the metric, i.e. whether length one represents one meter, one centimeter or one foot. In most context related to diffusion tensor imaging one simply speaks of the trace of the tensor, indirectly assuming that the tensor is expressed in an ON-basis for which the metric tensor is the identity matrix.

If the eigenvalue equation

$$D^{ij}x^j = \lambda x^i \quad (4.6)$$

has $n = \text{dim} V$ non-trivial solutions with corresponding linearly independent eigenvectors $e_i$ with eigenvalues $\lambda_i$, the matrix $D^{ij}$ may be decomposed according to the eigen decomposition theorem as

$$D^{ij} = (PW^{-1})^{ij} \quad (4.7)$$

where $P = [e^1, e^2, \ldots, e^n]$, $W^j_i = \lambda_i$ if $i = j$ and $W^j_i = 0$ if $i \neq j$. The eigenvalues may be found by solving the so-called characteristic equation

$$\begin{vmatrix}
D^1_1 - \lambda & D^1_2 & D^1_3 \\
D^2_1 & D^2_2 - \lambda & D^2_3 \\
D^3_1 & D^3_2 & D^3_3 - \lambda
\end{vmatrix} = 0 \quad (4.8)$$

equivalent to

$$A_1 = D^1_1 + D^2_2 + D^3_3 \quad (4.9)$$

$$A_2 = \begin{vmatrix}
D^2_2 & D^3_2 \\
D^3_3 & D^3_3
\end{vmatrix} + \begin{vmatrix}
D^1_1 & D^2_1 \\
D^1_2 & D^2_2
\end{vmatrix} + \begin{vmatrix}
D^1_1 & D^3_1 \\
D^1_3 & D^3_3
\end{vmatrix} \quad (4.10)$$

$$A_3 = \begin{vmatrix}
D^1_1 & D^1_2 & D^1_3 \\
D^2_1 & D^2_2 & D^2_3 \\
D^3_1 & D^3_2 & D^3_3
\end{vmatrix} \quad (4.11)$$

$$\lambda^3 - \lambda^2 A_1 + \lambda A_2 - \lambda A_3 = 0 \quad (4.12)$$
Any invariant which is independent of coordinate system may be written as a function of \( A_1, A_2 \) and \( A_3 \). The left hand side of the last equation is called the characteristic polynomial. Eigenvalues are independent of the choice of coordinate system and for this reason the coefficients in the polynomial are invariant to coordinate changes as well.

### Fractional anisotropy

The fractional anisotropy (FA) is a measure explaining how much the norm of the tensor stems from anisotropic contributions.

\[
FA = \frac{1}{\sqrt{2}} \frac{\sqrt{(\lambda_1 - \lambda_2)^2 + (\lambda_2 - \lambda_3)^2 + (\lambda_1 - \lambda_3)^2}}{\sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}} \quad (4.13)
\]

Due to the properties of the norm and the trace, it is invariant to rotations and scaling. See figure 4.5 for a typical axial slice displayed using \( FA \).

### Linear, planar & spherical

In (Westin et al., 2002) the following three measures of diffusion tensor shape are defined, corresponding to linear, planar and spherical shape

\[
c_t = \frac{\lambda_1 - \lambda_2}{\lambda_1} \quad (4.15)
\]

\[
c_p = \frac{\lambda_2 - \lambda_3}{\lambda_1} \quad (4.16)
\]

\[
c_s = \frac{\lambda_3}{\lambda_1} \quad (4.17)
\]

See figure 4.2 for an intuitive explanation of the concept.
4.2.2 Fiber tracking

While scalar invariants have been used widely, both to visualize and obtain quantitative measures of diffusion within the human brain, even more stunning visualizations and analyses of connectivity may be performed using so-called fiber tracking algorithms. They release seeds, virtual particles, in the data volume, creating streamlines while following the principal direction of diffusion (PDD). The tracking is usually seeded within white matter and terminates when reaching a gray matter mask or when the FA value becomes too low. See figure 4.6 for an example of fiber tracking.

**PDD tracking**

The simplest and maybe most widely used kind of fiber tracking is to follow the principal direction of diffusion. Each particle, seeded within white matter, is iteratively propagated along the principal direction of diffusion in the data. Great care should be taken in order to interpolate the tensor field within each voxel in order to obtain smooth fiber traces.

**Stochastic tracking**

In stochastic or probabilistic fiber tracking (Brun et al., 2002; Björnemo et al., 2002; Behrens et al., 2003b; Hagmann et al., 2003; Behrens, 2004; Behrens et al., 2003a; Friman et al., 2006), particles are propagated in a similar way as in PDD tracking. For each time step, a particle is propagated in a direction taken as a random sample from the estimated probability distribution of the PDD. In this way, uncertainty from the measurements and the model is taken into account. Seeding from a particular voxel A, multiple fiber traces are possible, and a kind of “connectivity estimate” \( p(B|A, t) \) may be calculated to measure the proportion of particles starting in a point A and reaching a point B after \( t \) time steps.

4.2.3 Fiber tract connectivity

Estimation of “connectivity” in the human has been something of a holy grail for the DT-MRI imaging community. Figures 4.3 and 4.4 show a dissection of a real brain, revealing some of the complexity of the human brain white matter architecture. If one can see fiber traces and fiber bundles in DT-MRI and in dissections of real brains, extending the algorithms to give a quantitative measure of connectivity ought to be possible. The probabilistic and stochastic algorithms for fiber tracking give quantitative answers to the question: \( p(B|A) \) = “what are the chances of ending of in voxel B if we start in voxel A” but this measure is not the same as \( p(A|B) \) which is a somewhat confusing property. Sometimes the connectivity measure is simply made symmetrical by brute force, i.e. \( c(A,B) = \frac{1}{2}(p(A|B) + p(B|A)) \) (Behrens, 2004).
4.2 Processing diffusion tensor data

One way to obtain a symmetric measure of connectivity would be to embed all voxels in a metric space (or even a manifold) in which a short (geodesic) distance \( d(A, B) \) means that two points A and B are more connected. In for instance (O’Donnell et al., 2002) the image volume is embedded by warping the metric according to the inverse of diffusion tensors. A problem with this approach could be that the triangle inequality plays a trick. Assume we have three points A, B and C in the brain. A is connected to B and A is also functionally connected to C. However, B and C are not connected at all. The triangle inequality says that \( d(B, C) \leq d(A, B) + d(A, C) \) and thus forces the points B and C to be close if A is connected to both B and C.

Apparently some work remains to be done before everybody agree on what kinds of anatomical connectivity there are, to what extent these quantities are possible to measure in DT-MRI and what the exact axiomatic properties, in a mathematical sense, should be for the various kinds of connectivity.
4.2.4 Segmentation of white matter

Without diffusion-weighted imaging, it is difficult to segment fiber bundles in human brain white matter. In other image modalities, voxels within white matter are represented by a single intensity and there is no way to distinguish between different bundles. With DT-MRI on the other hand, voxels in white matter may be segmented depending on what areas of the brain they connect. The same technique also works for segmenting gray matter into areas related to function (Behrens et al., 2003a).

Virtual dissection (Catani et al., 2002) is one example of how a medical doctor can interactively explore the anatomy of white matter by selecting fiber traces of interest depending on their connectivity. Other examples include automatic Fuzzy C-means (Shimony et al., 2002) clustering and NCut clustering (Brun et al., 2004) of DT-MRI fiber traces.

4.3 Visualization of streamline data

In a previous approach for visualization of DT-MRI data, presented in (Brun et al., 2004), methods inspired by dimension reduction and manifold learning are used in order to enhance the perception of connectivity in DT-MRI data of the human brain. This is different from obtaining quantitative measurements of connectivity and we envision these approaches to be useful for the purpose of interactive visualization and explorative analysis of DT-MRI. The primary goal is to create a visual interface to a complex dataset.

4.3.1 Local and global features in DT-MRI

The scalar invariants presented in 4.2.1 are important features of the kind of tensor-valued data obtained from DT-MRI. Using scalar invariants, features of the data inside a single voxel may be visualized using for instance a color map. This is an example of a local feature of the dataset. Another local feature in tensor data is edge information, see for instance (O’Donnell et al., 2004; Granlund and Knutsson, 1995). For vector-valued velocity data, which is also a kind of tensor data, features based on vortex and convergence/divergence have been proposed (Heiberg, 2001).

Connectivity as a feature

The connectivity of a voxel, for instance defined by streamlines or probabilistic fiber tracking, may also be regarded as a feature of that voxel. This not a local feature, while the connectivity of one single voxel depends on a spatially distributed set of voxels within the dataset. We call this a macro-feature. Voxels with a similar
connectivity profile may be mapped to similar places in a feature space describing
connectivity.

Viewing voxels as the atomic unit when visualizing connectivity in DT-MRI is
one alternative. The other alternative is to visualize streamlines. The main differ-
ence is that a streamline is itself a representation of its connectivity. A streamline
also has a simpler connectivity profile, while it connects exactly two endpoints
with each other. A single voxel on the other hand may, through for instance prob-
abilistic fiber tracking, connect to multiple endpoints. One single voxel may also
contain several – perhaps crossing – streamlines. This is particularly true if the
tracking algorithm or the data is rich enough to cope with crossing fiber bundles.

The shape and position of a streamline reveals its connectivity and in a way, also
the connectivity of the voxels it goes through. Similar streamlines usually belong
to the same fiber bundle

The fiber bundle assumption

Performing fiber tracking can be seen as a kind of feature transform, where the
data volume is transformed into a set of feature points. Each voxel inside the white
matter in the brain is used for seeding a fiber tracking procedure or performing
stochastic fiber tracking. The result is similar to a Hough transform, where each
fiber trace is analogous to the line integral of the Hough transform and maps to a
specific point in a fiber feature space.

In this fiber feature space, we assume there are clusters of points, correponding
to major fiber tracts such as the corpus callosum and the cingulum bundles. These
clusters of points live in a high-dimensional space, the fiber feature space, but
will intrinsically have only two dimensions corresponding to the cross section of
a fiber bundle. Early work on a similar topic may be found in (Westin, 1991).

4.3.2 Visualization of fiber tract connectivity

Scalar invariants

Using the scalar invariants defined in 4.2.1 we may visualize a 2-D slice of a 3-D
DT-MRI volume of a human brain. See figure 4.5 for a demonstration of fractional
anisotropy.

Glyphs

If the (2,0) or contravariant diffusion tensor is transformed into a (1,1) mixed
tensor using the metric $g_{ij}$, it is possible to interpret it as a linear transformation
and a spectral decomposition into eigenvectors and eigenvalues is possible.
Figure 4.5: An axial slice of a brain. **Left**: Intensity corresponds to fractional anisotropy. **Middle**: Color corresponds to main principal direction of diffusion. Red: left–right, green: anterior-posterior, blue: superior–inferior. **Right**: A rendering using tensor ellipsoid glyphs. Courtesy of Gordon Kindlmann.

Figure 4.6: A detail of an axial slice of the brain shown in figure 4.5. **Left**: Tensor ellipsoids. **Middle**: Tensor superquadrics (Kindlmann, 2004). **Right**: Streamtubes. Courtesy of Gordon Kindlmann.

Figure 4.7: An example of how techniques inspired by manifold learning and dimension reduction can be used to color fiber traces derived from diffusion tensor MRI, from (Brun et al., 2004).
In figure 4.6, two variants of tensor glyphs are shown: Ellipsoids and superquadrics (Kindlmann, 2004). Tensor glyphs show the strength, anisotropy and orientation of the diffusion tensors.

**Streamlines and streamtubes**

The result of fiber tracking may be visualized using either streamlines or streamtubes. By choosing the appropriate viewpoint, lighting and possibly a selection of a subset of fiber traces to visualize, it is possible to understand the geometry and connectivity of the dataset. See figure 4.6.

**Streamtube coloring**

When the set of fiber traces becomes too complex, an enhancement of the perception of connectivity may be created if the fiber traces are colored according to their position, shape and connectivity. Similar colors help the user to mentally group fiber traces into bundles. Fiber traces may also be clustered and colored in very different colors, see Fig. 4.7, to emphasize the difference between distinct clusters (Brun et al., 2004), for instance using dimension reduction and clustering algorithms inspired by manifold learning.
In this chapter, new results are presented for empirical LogMaps, a recently proposed framework for manifold learning and non-linear dimension reduction. Empirical LogMaps calculates the logarithmic map, i.e. the inverse of the exponential map, from a set of points sampled from a manifold embedded in a possibly high-dimensional Euclidean space. In contrast to most methods for manifold learning, this method has a very strong connection to basic differential geometry where \( \exp_p(x) \) and \( \log_p(x) \) are fundamental tools for the analysis of manifolds.

While the concept of manifolds and log maps are well known and well defined in mathematics, the estimation of models for manifolds and log maps from sampled data is an active area of research with several unsolved problems.

### 5.1 Introduction

With Antarctica being the exception, most of the world is visible in the map projection used in the emblem of the United Nations. It is also the most well known example of the so-called azimuthal equidistant map projection. This projection technique preserves angles and geodesic distances, from the center point on the North Pole to points in all directions up to a certain radius, making it useful to both radio amateurs and airline travelers. This mapping is a special case of the log map, \( \log_p(x) \), defined on a surface embedded in space, or more generally an \( m \)-dimensional manifold. This mapping, together with the exponential map, is depicted in Fig. 5.1. Note that \( x \in T_p M \), i.e. it is a vector in the tangent space to \( p \), while \( x \in M \), is a point in the manifold.

The log map was first proposed for use in manifold learning by Brun et al. (2005), who derived a method to estimate the log map from a set of data points sampled from a manifold. It was based on numerical estimation of gradients to an estimated geodesic distance function. While the actual manifold is only known from samples, the estimated log map is different from the real log map. We call this class of methods empirical LogMaps, sample LogMaps or LogMaps for short.
Chapter 5. Empirical LogMaps

![Diagram](image)

**Figure 5.1:** The log function is the inverse of exp function. While exp maps vectors in the tangent plane in \( p \) to points on the surface, the log function maps points on the surface back to the tangent plane. Riemannian Normal Coordinate are obtained by expressing points in the tangent plane at \( p \) in a ON basis and thus the \( \log \) function is able to map points on the surface, via tangent vectors, to (texture) coordinates.

For the users of manifold learning, there already exist many mappings to choose from. One may ask what makes LogMaps special when compared to Isomap, PCA and other possible mappings from points sampled from a manifold embedded in a vector space? The log map defines a coordinate chart to a manifold, assigning \( m \)-dimensional coordinates to points in an \( m \)-dimensional manifold. These coordinates are called Riemannian Normal Coordinates and are popular in differential geometry due to their special properties. In a sense, \( \log_{p}(x) \) is the “most linear” mapping near \( p \), because geodesics or length minimizing curves, emanating from \( p \), are mapped to straight lines in the vector space spanned by the coordinates.

The log map, accompanied by \( \exp_{p}(x) \), is also a fundamental building block in several frameworks for programming in manifolds and signal processing of manifold-valued signals. In combination with interesting computational properties, this makes LogMaps promising candidates for the future.

### 5.2 Related work

Geometric frameworks for learning and dimension reduction have been around for more than a century, Principal Component Analysis (PCA) being the earliest example (Pearson, 1901). Development of efficient methods to model non-linear relations in data has progressed in several steps, Self Organizing Maps (SOM) (Kohonen, 1982) and Kernel Principal Component Analysis (KPCA) (Schölkopf et al., 1998) both represent major breakthroughs. The most recent line of research in this field, often based on the solution of large eigenvalue problems and called “manifold learning”, started with Isomap (Tenenbaum et al., 2000) and LLE (Roweis and Saul, 2000) and is still an active area of research. For all of the above examples, the mapping is found as a result of a procedure optimizing a specific criterion, such as maximization of variance in the case of PCA. In most
5.2 Related work

<table>
<thead>
<tr>
<th>Operation</th>
<th>Vector space equation</th>
<th>Manifold equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subtraction</td>
<td>$x \bar{y} = y - x$</td>
<td>$\bar{x}y = \log_x(y)$</td>
</tr>
<tr>
<td>Addition</td>
<td>$y = x + \bar{y}x$</td>
<td>$y = \exp_x(\bar{x}y)$</td>
</tr>
<tr>
<td>Distance</td>
<td>$\text{dist}(x, y) =</td>
<td></td>
</tr>
<tr>
<td>Mean value (implicit)</td>
<td>$\sum_i \bar{x}x_i = 0$</td>
<td>$\sum_i \log_x(x_i) = 0$</td>
</tr>
<tr>
<td>Gradient descent</td>
<td>$x_{t+\epsilon} = x_t - \epsilon \nabla C(x_t)$</td>
<td>$x_{t+\epsilon} = \exp_{x_t}(-\epsilon \nabla C(x_t))$</td>
</tr>
<tr>
<td>Linear interpolation</td>
<td>$x(t) = x_1 + t\bar{x}_1\bar{x}_2$</td>
<td>$x(t) = \exp_{x_t}(t\bar{x}_1\bar{x}_2)$</td>
</tr>
</tbody>
</table>

Table 5.1: In (Pennec et al., 2005) the above analogies are made between operations in vector spaces and manifolds.

modern approaches to manifold learning and dimension reduction, this criterion is designed to yield a convex objective function, for which a global optimum can be found using standard optimization methods.

5.2.1 Programming on manifolds

One goal in manifold-valued signal processing is to represent the signal processing algorithms in a coordinate free way. This means that the operations have a geometric, intrinsic meaning, not relying on any particular coordinate system. For example, using coordinate free methods, operations can be defined on the whole of $S^2$ while any coordinate description must have coordinate singularities. In a way, coordinate free approaches actually point towards not using manifold learning to find a low-dimensional parameterization of the data manifold, but instead perform all data- and signal processing intrinsically in the manifold. As pointed out in (Pennec et al., 2005)

“the implementation of $\log_x$ and $\exp_x$ is the basis of any programming on Riemannian manifolds”.

Using LogMaps we now have a basic building block for performing some of these calculations for signal and data processing in sampled manifolds. In table 5.1 that is reproduced from (Pennec et al., 2005), some basic operations in vector spaces are compared to analogous operations in manifolds. In fact, closed form expressions for $\log_p(x)$ in Lie groups have already been used for medical shape analysis (Fletcher et al., 2004) and we believe that any LogMap could potentially be plugged in to this framework.

The fact that $\log_p(x)$ is used in other algorithms, when known analytically, is one motivation to why LogMaps and RNC could be better suited for certain tasks than mappings defined by other methods for manifold learning.

5.2.2 Previous work on Riemannian normal coordinates

Brun et al. (2005) were the first to show how $\log_p(x)$ can be found from a set of points sampled from a manifold and related this procedure to manifold learning.
The method was based on the geodesic distance estimation technique from the Isomap algorithm (Tenenbaum et al., 2000), which uses Dijkstra’s algorithm for shortest paths in graphs to estimate geodesic distances in a sampled manifold. The zigzag artifacts from this distance estimate affect the efficiency of the original LogMap. Used with more accurate distance estimation, e.g. based on higher order schemes compared to the nearest neighbor interpolation used in Isomap, this original LogMap can be both accurate and fast, which is also shown later in chapter 6.

The original LogMap approach was evaluated by Kayo (2006). Later Lin et al. (2006) proposed an alternative method to directly estimate a LogMap from the samples, as a part of the Dijkstra-loop. This approach is claimed to be faster and certainly works for flat manifolds and it would be interesting to compare this method to the LogMap framework in the future.

5.3 The LogMap algorithm

The function $\log_p(x)$ in a manifold is a mathematically well-defined function, which maps points $x$ on the manifold to the tangent space in $p$, $T_p M$. It is the inverse of the exponential function, $\exp_p(x)$, which maps a vector $x \in T_p M$ to points on the manifold, see Fig. 5.3.

The steps in the original LogMap algorithm (Brun et al., 2005) for a single point $y$ are as follows. See also figure 5.2.

1 Define $N(p)$ as the ball of $k$ closest points around $p$.

2 Calculate the coordinates of all points in $N(p)$ in an ON-basis.
3 Estimate the distances from a point $y$ in the manifold to all points in $N(p)$.

4 Estimate the log map by numerical approximation of the gradient of the squared distance function, see equation 5.3 below for the exact expression.

Another hint on how $\log_p(x)$ can be computed numerically is to consider some results related to how the so-called intrinsic mean is computed (Karcher, 1977; Fletcher et al., 2004). Let $\{x_i\}$ be $N$ data points in a manifold $M$ and seek the minimizer to the function

$$f(p) = \frac{1}{2N} \sum_{i=1}^{N} d^2(p, x_i),$$

(5.1)

where $d^2(p, x_i)$ is the squared geodesic distance between points $p$ and $x_i$. It is shown in (Karcher, 1977) that, under the appropriate assumptions of convexity, the gradient of $f$ is

$$\nabla f(p) = -g_{st} \frac{1}{N} \sum_{i=1}^{N} \log_p x_i.$$  

(5.2)

Setting $N = 1$ and $x_1 = x$ gives the following formula,

$$\log_p(x) = -g_{st} \frac{1}{2} \nabla_y d^2(y, x) \bigg|_{y=p}.$$  

(5.3)

The metric $g_{st}$ and inverse metric $g^{st} = (g^{-1})^{st}$ have been added here to handle the general case, but choosing an ON-basis for $T_pM$ yield $g_{st} = g^{st} = \delta_{st}$ and allow us to identify co- and contravariant vectors. This is of course a formula only related to the true mathematical log map. In order to obtain a LogMap based on samples, both the differentiation and the distance estimation must be evaluated numerically.

The general algorithm of the LogMap framework is presented in Alg. 2. It is fairly abstract and does for instance not point out how the user should select a suitable coordinate system. Its main purpose is to define exactly what a LogMap algorithm is. A less general algorithm that is a lot more straightforward to implement is presented in Alg. 4. It is the algorithm that has been used in most of the experiments. For the convenience of the reader we have also included the classic MDS algorithm in Alg. 3.

In the following section we will briefly mention some mathematical properties for Riemannian Normal Coordinates and LogMaps.
Chapter 5. Empirical LogMaps

Algorithm 2 LogMap: General estimate of $\mathbf{x} = \log_p(x)$

Require: $p$ and $x$ are points in the Riemannian manifold $M$, $\dim(M) = m$.

$X = \{x_1, x_2, \ldots, x_N\}, x_i \in M$, are samples from $M$. $N(p) \subset M$ is a neighborhood to $p$.

A procedure $d_X : M \times M \rightarrow \mathbb{R}$ estimates geodesic distances in $M$ using samples in $X$.

Ensure: $\hat{x} \in T_pM$, estimate of $\log_p(x_i)$.

1: [Local Parameterization] Define $\varphi : N(p) \rightarrow S \subset \mathbb{R}^m$, a local coordinate system or chart, and a metric $G_{ij}(y) : T_yM^* \times T_yM^* \rightarrow \mathbb{R}$ is defined on $y \in N(p)$.

2: [Distance Estimation] Estimate geodesic distances $r(y) = d_X(x, y)$ from $x$ to all samples $y \in N(p) \cap X$.

3: [Function Approximation] Find an approximation $h(z)$ to the squared geodesic distance function $r^2(\varphi^{-1}(z))$, $z \in S$, using function approximation and estimates calculated in the previous step for all $z = \varphi(y), y \in N(p) \cap X$.

4: [Gradient Calculation] Calculate estimate of $\log_p(x)$, $\hat{x} = -\frac{1}{2}(G_{ij}(z))^{-1}(\nabla h(z))^T|_{z=\varphi(p)}$, where $\nabla$ is the partial derivative, or covariant derivative, written as a row vector.

Algorithm 3 Classical Multidimensional Scaling (MDS)

Require: $D_{ij} = d^2(y_i, y_j)$, a matrix of pairwise distances between elements in some unknown vector space. A parameter $m$ determine the dimensionality of the output vectors.

Ensure: The algorithm finds an optimal set of vectors, $Z = (z_1, z_2, \ldots, z_k)$, $z_i \in \mathbb{R}^m$, for which $d(z_i, z_j) \approx D_{ij}$, i.e. $Z = \arg \min_Z \sum_{i=1}^k (d^2(y_i, y_j) - ||z_i - z_j||^2)^2$.

1: Form a kernel matrix, $K$,

\[ K_{ij} = -\frac{1}{2} \left( D_{ij}^2 - \frac{1}{k} \sum_i D_{ij}^2 - \frac{1}{k} \sum_j D_{ij}^2 + \frac{1}{k^2} \sum_{i,j} D_{ij}^2 \right). \]

2: Perform eigenvalue decomposition, $K = Q\Lambda Q^T$, where $Q^TQ = I$ and $\Lambda$ is diagonal with elements $\lambda_i$.

3: New coordinates of $Y$ are $Z = (z_1, z_2, \ldots, z_k)$, $z_{ij} = \sqrt{\lambda_i}Q_{ji}$, where $1 \leq i \leq m$ and $m$ is the dimensionality of the manifold. By construction $\sum_i z_i = 0$. 
Algorithm 4 LogMap, Simplified LogMap Estimation

Require: $M$ is a Riemannian manifold embedded in $\mathbb{R}^n$. $X = (x_1, x_2, \ldots, x_N)$, $x_i \in M$. The base point $p \in X$. $k$ is a parameter determining the neighborhood size used to estimate gradients. $M$ is a parameter determining the number of neighbors used in the geodesic distance estimation. The dimensionality of the manifold is assumed to be known and equal to $m$.

Ensure: $\hat{x} \in T_p M$, estimate of $\log_p(x_i)$.

1: [Local Parameterization] Let $Y = (y_1, y_2, \ldots, y_k)$, $y_i \in X$, the ball containing the $k$ closest points to $p$, including $p$, measured in the Euclidean metric of $\mathbb{R}^n$. Estimate Euclidean distances, $D_{ij} = ||y_i - y_j||_2$, and perform classical multidimensional scaling (MDS) to find new coordinates $z_i = \varphi(y_i)$, $z_i \in \mathbb{R}^m$. In this coordinate system we assume $G_{ij}(z) = I$.

2: [Distance Estimation] Let $d_X : X \times X \to \mathbb{R}$ be Dijkstra’s algorithm applied on the dataset $X$ as described in (Tenenbaum et al., 2000), with $M$ neighbors used to construct the graph. Estimate squared geodesic distances $\Delta_{ij} = d_X^2(y_i, x_j)$ for all $y_i \in Y$ to all $x_j \in X$.

3: [Function Approximation] For each $x_j \in X$, find second order approximation to the squared geodesic distance, $h_j(z_i) \approx d_X^2(y_i, x_j)$,

$$h_j(z) = a_j + b_j^T z + z^T C_j z.$$

The optimal coefficients, in a least squares sense, are found by

$$
\begin{pmatrix}
  a_j \\
  b_j(z) \\
  C_j(z)
\end{pmatrix} = \left( \begin{array}{ccc} 1 & z_1(:,)^T & (z_1 z_1^T)(:)^T \\
  1 & z_2(:,)^T & (z_2 z_2^T)(:)^T \\
  \vdots & \vdots & \vdots \\
  1 & z_k(:,)^T & (z_k z_k^T)(:)^T \\
\end{array} \right)^{\dagger} \left( \begin{array}{c} \Delta_{1j} \\
  \Delta_{2j} \\
  \vdots \\
  \Delta_{kj} \\
\end{array} \right),
$$

where $(:,)$ denotes vectorization into a column vector. This solution is efficient to compute for many points $x_j$, since the pseudo-inverse only needs to be evaluated once for each base point $p$.

4: [Gradient Calculation] Calculate estimate of $\log_p(x)$,

$$\dot{x}_j = -\frac{1}{2} \nabla h_j(z) \bigg|_{z=\varphi(p)} = -\frac{1}{2} b_j^T - \varphi(p)^T C_j \approx -\frac{1}{2} b_j^T,$$

if $\varphi(p)$, the position of $p$ in the local coordinate system, is approximated by $0$, which is often reasonable since the points in $Y$ were selected by a ball around $p$. 


5.4 Mathematical properties of RNC and LogMaps

Here we briefly state some interesting mathematical properties for Riemannian Normal Coordinates and the \( \log_p(x) \) mapping.

5.4.1 The LogMap formula

To prove the formula we use for the estimation of \( \log_p(x) \), we first provide some results related to RNC.

**Proposition 5.4.1 (Properties of Riemannian Normal Coordinates).** From (Lee et al., 1997). Let \((U, (x^i))\) be any normal coordinate chart centered at \( p \). The radial distance function is defined by

\[
r(x) = \left( \sum_i (x^i)^2 \right)^{1/2}.
\]

(a) For any \( V = V^i \partial_i \in T_p M \), the geodesic \( \gamma_V \) starting at \( p \) with initial velocity vector \( V \) is represented in Riemannian normal coordinates by the radial line segment

\[
\gamma_V(t) = (tV^1, \ldots, tV^n) \tag{5.4}
\]

as long as \( \gamma_V \) stays within \( U \).

(b) The coordinates of \( p \) are \((0, \ldots, 0)\).

(c) The components of the metric at \( p \) are \( g_{ij} = \delta_{ij} \).

(d) Any Euclidean ball \( \{x : r(x) < \epsilon\} \) contained in \( U \) is a geodesic ball in \( M \).

(e) At any point \( q \in U - p \), \( \partial / \partial r \) is the velocity vector of the unit speed geodesic from \( p \) to \( q \), and therefore has unit length with respect to \( g_{ij} \).

(f) The first partial derivatives of \( g_{ij} \) and the Christoffel symbols vanish at \( p \).

In the following theorem we prove Eq. 5.3.

**Theorem 5.4.1 (The LogMap formula).** Let \( r_q(p) = d(q, p) \) and \( \frac{\partial}{\partial r} = r_i g^{ij} \). Then

\[
\exp_p\left(-\frac{1}{2}r^2 g^{ij}\right) = q. \tag{5.5}
\]

**Proof.**

\[
\exp_p\left(-\frac{1}{2}r^2 g^{ij}\right) = q \iff \exp_p\left(-rr_i g^{ij}\right) = q \iff \exp_p\left(-r \frac{\partial}{\partial r}\right) = q \tag{5.6}
\]

From Prop. 5.4.1 we have that \( \frac{\partial}{\partial r} \) has unit length w.r.t. \( g_{ij} \) and points along radial geodesics from \( q \). Thus \( -r \frac{\partial}{\partial r} \) points along geodesics to \( q \) and since \( r = d(q, p) \), \( \exp_p\left(-r \frac{\partial}{\partial r}\right) = \gamma(1) \), where \( \gamma(t) \) is the geodesic emanating from \( p \) with speed \( -r \frac{\partial}{\partial r} \) after time \( t \). \( \square \)
5.4.2 On the optimality of LogMaps

One may ask why the log map is useful to perform dimension reduction and visualization for data points. Suppose we are looking for some mapping \( f : M \rightarrow \mathbb{R}^m \) such that

1. \( f(p) = 0 \),
2. \( d(p, x) = \|f(x)\| \),
3. \( d(y, x) \approx \|f(y) - f(x)\| \) when \( y \approx p \).

In short, this would be a mapping that preserves all distances exactly between points \( x \in M \) and the base point \( p \in M \). For points \( y \in M \) that are close to \( p \), distances are approximately preserved. It turns out that this mapping is the \( \log_p(x) \) mapping and it is expressed in the following theorem.

**Theorem 5.4.2.** Suppose \( f : M \rightarrow \mathbb{R}^n \) is a continuous mapping and \( f(p) = 0 \). Then

\[
d(x, y) = \|f(x) - f(y)\| + \|f(y)\|^2 B(x, y),
\]

for some bounded function \( B \), if an only if

\[
f(x) = A \log_p(x), \quad A \in O(n),
\]

where \( O(n) \) is the group of orthogonal transformations, \( A^T A = I \).

**Proof.**

\[
\exists B : d(x, y) = \|f(x) - f(y)\| + \|f(y)\|^2 B(x, y) \iff
\]

via Taylor approximation on both sides

\[
\exists B_1 : d(x, p) + \langle \nabla_p d(x, p), \log_p(y) \rangle =
\]

\[
\|f(x)\| - \frac{\langle f(x), f(y) \rangle}{\|f(x)\|} + B_1(x, y)\|f(y)\|^2 \iff
\]

\[
\exists B_2 : \frac{\langle \log_p(x), \log_p(y) \rangle}{d(x, p)} = \frac{\langle f(x), f(y) \rangle}{\|f(x)\|} + B_2(x, y)\|f(y)\|^2 \iff
\]

\[
\exists B_3 : \frac{\langle \log_p(x), \log_p(y) \rangle}{d(x, p)d(y, p)} = \frac{\langle f(x), f(y) \rangle}{\|f(x)\|\|f(y)\|} + B_3(x, y)\|f(y)\| \iff
\]

\[
f(x) = A \log_p(x), \quad A \in O(n),
\]

The \( \iff \) in the last step is obvious, while the \( \Rightarrow \) follows from the fact that the expression should be valid for \( y \) arbitrary close to the base point \( p \). \( \square \)

From this result we can state that LogMaps, or rather the true \( \log_p(x) \), is the optimal mapping in the above sense, i.e. it is the most linear mapping centered at the base point \( p \).
5.5 Experiments

The LogMap method was evaluated using MATLAB. The most critical part of the algorithm, the calculation of shortest paths, was borrowed from the Isomap implementation of Dijkstra’s shortest paths algorithm (Tenenbaum et al., 2000).

5.5.1 The Swiss roll

In the first experiment we use the “Swiss roll” data set, consisting of points sampled from a 2-D manifold, embedded in \( \mathbb{R}^3 \), which looks like a roll of Swiss cheese. It has been used before to illustrate methods for manifold learning, see e.g. (Tenenbaum et al., 2000; Roweis and Saul, 2000), and we include it mainly as a benchmark. A set of 5000 points from this data set were used in the experiment and the results are presented in figure 5.3. The experiment shows that the empirical LogMap method correctly unfolds the roll and maps it to Riemannian normal coordinates in \( \mathbb{R}^2 \).

5.5.2 The torus

In the second experiment we use a torus data set, consisting of points sampled from an ordinary torus embedded in \( \mathbb{R}^3 \). A set of 5000 points from this data set were used in the experiment and the results are presented in figure 5.4. The experiment demonstrates how the cut locus efficiently cuts the manifold at the antipodal point of \( p \). In this manner, this surface may be visualized in 2-D instead of 3-D, at the expense of a discontinuity in the mapping.

5.5.3 Local phase

In an experiment originating from (Brun et al., 2005) a set of small image patches with different phase and orientation were used as input for the LogMap algorithm. In Fig. 5.5 a regular sampling of this set of patches is shown. In Fig. 5.6 the original results from (Brun et al., 2005) have been reproduced, showing how the Klein bottle topology was discovered experimentally. In Fig. 5.7 additional experiments have been performed, showing the different mappings obtained by selecting different base points \( p \). This may seem strange at first, but in Fig. 5.8 it is explained why different base points generate different results. It is because the Klein bottle is, somewhat unexpectedly perhaps, not a globally symmetric manifold.

5.5.4 Blob-shapes

In a final example, a set of 1100 2-D images of blobs of varying shape has been created from random samples of a rectangular parameter space. See figure 5.9. We tried both the LogMap method and the well-known PCA on this dataset. The PCA method seems to have revealed the true parameter space, but close inspection
reveals that the mapping is both non-linear, w. r. t. the original parameter space, and suffers from aliasing (i.e. dissimilar shapes are mapped to the same position). The LogMap works fine in this example.
Figure 5.3: A set of 5000 points from the “Swiss roll” example (Tenenbaum et al., 2000). Colors correspond to the first Riemannian normal coordinate derived from the method. **Top:** The original point cloud embedded in 3-D. **Bottom:** Points mapped to 2-D Riemannian normal coordinates.
Figure 5.4: A set of 5000 points from a torus. Colors correspond to the first Riemannian normal coordinate derived from the method. **Top:** The original point cloud embedded in 3-D. **Bottom:** Points mapped to 2-D Riemannian normal coordinates.
Figure 5.5: An artificially generated map sampling the set of all small image patches with different phase and an orientation. Phase varies in the horizontal direction and orientation varies in the vertical direction.
5.5 Experiments

Figure 5.6: From (Brun et al., 2005). To test the proposed method on a high-dimensional dataset, a set of 900 image patches, each being of $21 \times 21$ pixels with a characteristic orientation/phase, were generated and mapped to Riemannian normal coordinates. This experiment reveals the Klein bottle-structure of local orientation/phase in 2-D image patches. **Top-Left**: An idealized Klein bottle aligned to the mapping below. Edges correspond to the cut locus of $p$ and should be identified according to the arrows. **Top-Middle**: An immersion of the Klein bottle in 3-D. **Top-Right**: 15 random examples of image patches used in the experiment. **Bottom**: The mapping of image patches to Riemannian normal coordinates using the proposed method.
Figure 5.7: Four additional results from applying LogMaps to different base points $p$ in the space of all image patches. This experiment demonstrates that the Klein bottle is not globally symmetric, since some base points generate charts with a different border.

5.5.5 Conclusion

We have presented additional theory for LogMaps, connecting them stronger to differential geometry. Some motivations to what is special with LogMaps have also been provided. Finally some experiments were provided to illustrate the power of the framework.
Figure 5.8: Equivalence classes of points in the plane can represent points in a Klein bottle. A point \( p \) inside the square \( abcd \) is repeated over the plane according to the symmetry shown in the graph above and is thereby represented by a set. Intrinsic distances between two points \( p \) and \( q \) is defined as the minimum distance between the set of \( p \) and \( q \). The point \( p \) represented by a small circle in the figure is surrounded by a polygonal set which represent points for which \( p \) is the closest point in this metric. This corresponds to the log map of \( p \) in the Klein bottle and the border of this polygon is the so-called cut locus. In the figure this polygon has 6 edges, but depending on the choice of \( p \) its shape varies, showing that a Klein bottle is not globally symmetric even though it is flat.
Figure 5.9: Data consists of 1100 images, each with a resolution of $67 \times 67$ pixels. Each image contains a white blob of varying shape. 100 of the data points were regularly sampled in a $10 \times 10$ grid in the parameter space. **Top:** An example of the empirical LogMap. The center point $p$ is positioned in $(0, 0)$ and $K = M = 50$. **Bottom:** The result of running PCA on the same data set.
This chapter presents a framework for the calculation of Riemannian Normal Coordinates (RNC) on a triangular surface mesh. It utilizes accurate geodesic distance information from an eikonal solver based on a newly introduced iterative method, which in combination with the LogMap method enables calculation of RNC centered at an arbitrary point $p$. This coordinate system has previously been used in the computer graphics literature for local texture mapping. Because of the close connection between RNC and the exponential map on surfaces and manifolds in general, this parameterization is also well motivated from a mathematical and differential geometry point of view. The framework we present is general and potentially more accurate than previous methods.

### 6.1 Introduction

In this chapter we demonstrate a framework for texture mapping which we believe is conceptually simpler and more general than a recently proposed method called ExpMap (Schmidt et al., 2006). Given accurate geodesic distance estimates on the surface, generated by any geodesic distance estimation algorithm, it is able to generate texture coordinates by a very simple procedure which is related to the LogMap framework presented in chapter 5.

### 6.2 Previous work

A lot of research has been devoted to texture mapping and finding suitable texture coordinates on a triangulated mesh. Optimization for minimal area/angle distortion has been a key instrument to evaluate the generated parameterizations. While perceptual measures has been important for some applications in computer graphics, the application of purely mathematical concepts has also played an important role, such as the use of conformal maps (Angenent et al., 2000) which preserve angles locally but distort local area.
Chapter 6. LogMap texture mapping

Figure 6.1: **Left**: How LogMap works. Distances (dashed iso-lines) are calculated from all \( k \) points in a neighborhood to the base point \( p \), to all other points \( q \) in the surface. **Right**: How LogMap works. The gradient of the distance function (dashed iso-lines) within the neighborhood to \( p \) are used to calculate the direction and distance (arrow) to \( q \) along a geodesic (solid curve segment) from \( p \).

The technique we present and review here is related to both differential geometry and manifold learning. The latter is a recent branch of unsupervised machine learning. In (Brun et al., 2005) the LogMap method was originally presented as a means to estimate Riemannian Normal Coordinates for points in a manifold known only from an unordered set of samples embedded in an Euclidean space. Unlike the triangular meshes in this chapter, the topology was not known beforehand, and the resulting mappings had relatively large numerical errors for small sample sizes since the geodesic distances were calculated using Dijkstra’s algorithm, inspired by (Tenenbaum et al., 2000). In a related paper (Lin et al., 2006), also devoted to the estimation of Riemannian Normal Coordinates, some of these problems were addressed.

Within computer graphics, a recently proposed method called ExpMap (Schmidt et al., 2006) has been presented as a tool to map small texture decals to triangular meshes and perform texture mapping of developable surfaces in general. Despite the name, it appears that ExpMap aim at computing estimates of \( \log_p(x) \), i.e. maps vertices on the surface to texture coordinates. However, some numerical approximations in the ExpMap algorithm are specifically tailored for developable surfaces (planes, cylinders, cones, . . . ) and suggest that it will not yield the true log map for non-developable surfaces (spheres, a plane with a bump, . . . ). In the paper (Schmidt et al., 2006), is argued that this behavior is not a bug but instead a feature in some situations. Nevertheless, ExpMap was the main motivation for testing LogMap in computer graphics.
6.3 The LogMap method

The original LogMap algorithm (Brun et al., 2005) was reviewed in the previous chapter. In Fig. 6.1 the algorithm is also explained in the setting of 2-D surfaces embedded in 3-D. The exact formula is given by

$$\log_p(x) = -\frac{1}{2} \nabla_y d^2(y, x) \bigg|_{y=p},$$

(6.1)

where $\nabla_y$ means the gradient with respect to $y$ and $d$ is the geodesic distance function. To get some intuition for this formula, we may test it in the plane with the trivial geodesic distance function $d(x, y) = ||x - y||_2$ and $p = 0$. Then

$$\log_p(x) = -\frac{1}{2} \frac{\partial}{\partial y} (x - y)^T (x - y) \bigg|_{y=0} = x,$$

(6.2)

which is the identity mapping. Although this is a trivial example, we would like to stress that the coordinates of the geometrical vector $x$ actually depend on which coordinate system is being used. In particular, it is safe to replace the gradient, $\nabla$, with partial derivatives if we are using an ON coordinate system, but it is not correct if the coordinate system is not ON. Depending on the application, the Cartesian RNC coordinates may be converted to polar. In polar coordinates, isolines of the angular coordinate will be aligned with geodesics.

An algorithm for the estimation of $\log_p(x)$ in triangular meshes is described in Alg. 5. In essence, the LogMap method is gradient estimation. Given the distance maps from seed points close to $p$, the LogMap method only adds the complexity of some matrix multiplication per vertex in the mesh. The pseudoinverse only needs to be evaluated once. If the distance estimation has errors, a larger number of seed points will result in a smoother map because the errors are averaged. For LogMap on surfaces however, three seed points ($k = 3$) is theoretically enough.

6.4 Computing geodesic distance

The eikonal equation governing geodesic distance has important applications in many fields; it is used in seismology, optics, computer vision (Bruss, 1989), and medical imaging and image analysis. It is also frequently used in computer graphics, for example the level set method (Osher and Sethian, 1988) makes heavy use of the eikonal equation to condition the embedding scalar field.

This special case of the Hamilton-Jacobi non-linear partial differential equation

$$|\nabla u(x)| = F(x), \; x \in \Omega$$

(6.3)

with initial conditions $u|_{\partial \Omega} = 0$ can be further simplified to describe isotropic uniform geodesic distance by setting $F(x) = 1$. Intuitively this means that any
Algorithm 5 LogMap for triangular surface meshes

Require: $M$ is a surface embedded in $\mathbb{R}^3$, $X = (x_1, x_2, \ldots, x_N)$, $x_i \in M \subset \mathbb{R}^3$ are the vertices. The base point is $p \in X$. $k$ is a parameter determining the neighborhood size used to estimate gradients.

Ensure: $\hat{x} \in T_p M$, estimate of $\log_p(x_i)$.

1: [Local Parameterization] Let $Y = (y_1, y_2, \ldots, y_k)$, $y_i \in X$, the ball containing the $k$ closest points to $p$, including $p$, measured in the Euclidean metric of $\mathbb{R}^3$. Calculate Euclidean distances, $D_{ij} = ||y_i - y_j||_2$, and perform classical multidimensional scaling (MDS) to find new coordinates $z_i = \varphi(y_i)$, $z_i \in \mathbb{R}^2$. Or define a ON coordinate system of choice.

2: [Distance Estimation] Let $d_X : X \times X \rightarrow \mathbb{R}$ be the geodesic distance algorithm applied on the dataset $X$ and estimate squared geodesic distances, $\Delta_{ij} = d_X^2(y_i, x_j)$, for all $y_i \in Y$ to all $x_j \in X$.

3: [Function Approximation] For each $x_j \in X$, find second order approximation to the squared geodesic distance for all points in $Y$, $h_j(z) = h_j(\varphi(y_i)) \approx d_X^2(y_i, x_j)$.

$$h_j(z) = a_j + b_j^T z + z^T C_j z.$$ 

The optimal coefficients $a_j$, $b_j(\cdot)$ and $C_j(\cdot, k)$, in a least squares sense, are found by

$$\begin{bmatrix} a_j \\ b_j(\cdot) \\ C_j(\cdot) \end{bmatrix} = \begin{bmatrix} 1 & z_1(\cdot)^T (z_1 z_1^T)(\cdot)^T \\ 1 & z_2(\cdot)^T (z_2 z_2^T)(\cdot)^T \\ \vdots & \vdots & \vdots \\ 1 & z_k(\cdot)^T (z_k z_k^T)(\cdot)^T \end{bmatrix}^\dagger \begin{bmatrix} \Delta_{1j} \\ \Delta_{2j} \\ \vdots \\ \Delta_{kj} \end{bmatrix}$$

where $(\cdot)$ denotes vectorization into a column vector. This step is efficient since the pseudo-inverse $(\dagger)$ only needs to be evaluated once.

4: [Gradient Calculation] Estimate $\hat{z}_j = \log_p(x_j)$ by

$$\hat{z}_j = -\frac{1}{2} \nabla h_j(z)^T \bigg|_{z = \varphi(p)} = -\frac{1}{2} b_j^T - \varphi(p)^T C_j \approx -\frac{1}{2} b_j^T$$

if $\varphi(p)$, the position of $p$ in the local coordinate system, is approximated by 0, which is often reasonable since the points in $Y$ were selected by a ball around $p$. 


function or field that fulfills Eq. 6.3 has unit gradient pointing in the direction of steepest ascent, namely along the path of geodesics. Thus, any geodesic path through the field is guaranteed to have uniform slope.

Usually Eq. 6.3 would be solved using a monotonic update scheme, such as Sethian’s fast marching method (Sethian, 2001). A recent new method (Jeong and Whitaker, 2007) with promising results also does a good survey of existing algorithms. When used on triangular meshes these methods, although first order accurate, assume a linear interpolation of distance values along edges, which introduces an error as large as 20% for some updates depending on triangulation. They use a linear edge interpolant, uniquely defined by two vertices $v_j$ and $v_k$, defined as

$$F_l(e_{jk}) = u((1-t)v_j + tv_k) = (1-t) \cdot u(v_j) + t \cdot u(v_k) \quad t \in [0, 1]$$  \hspace{1cm} (6.4)

In (Reimers, 2004), a non-linear interpolant that is exact in the plane is introduced, which also uses information about the distance between $v_j$ and $v_k$,

$$F_{nl}(e_{jk}) = \sqrt{(1-t)^2u_j^2 - t(1-t)||v_j - v_k||^2 + tu_k^2}.$$  \hspace{1cm} (6.5)

Solving the eikonal equation using this interpolant, however, needs careful consideration since Eq. 6.5 is not monotonic by construction. An update scheme with practical running times comparable with the Fast Marching Algorithm is presented in the paper and reproduced in Alg. 6 for convenience.

### 6.5 Experiments

The experiments were carried out in a mixed environment of code in MATLAB and C++.

#### 6.5.1 The Stanford bunny

The familiar Stanford bunny was used an experiment where the base point was placed on the back of the bunny. The resulting mappings were used for texture mapping, as seen in Fig. 6.2. In Fig. 6.3 we zoom in to the cut locus of the mapping, which has a very sharp appearance. The overall impression from this experiment is that the LogMap framework seems to work very well, given accurate distance estimates.
Algorithm 6 Euclidian distance computation

Require: Input source index \( s \)

\[
\begin{align*}
1: & \quad \text{for } i = 0, \ldots, N \text{ do} \\
2: & \quad \text{if } i \in s \text{ then} \\
3: & \quad \quad U[i] \leftarrow 0 \\
4: & \quad \text{else} \\
5: & \quad \quad U[i] \leftarrow \infty \\
6: & \quad \text{end if} \\
7: & \quad \text{end for} \\
8: & \quad \text{candidates.push(s)} \\
9: & \quad \text{while candidates.notEmpty() do} \\
10: & \quad \quad i = \text{candidates.getSmallestNode()} \\
11: & \quad \quad \text{for } j \in \text{dependents}(i) \text{ do} \\
12: & \quad \quad \quad \text{newVal} = \text{update}(j) \\
13: & \quad \quad \quad \text{if } \text{newVal} \leq U[j] + \epsilon \text{ then} \\
14: & \quad \quad \quad \quad U[j] = \text{newVal} \\
15: & \quad \quad \quad \quad \text{candidates.push(j)} \\
16: & \quad \quad \text{end if} \\
17: & \quad \quad \text{end for} \\
18: & \quad \text{end while}
\end{align*}
\]

6.5.2 Plane with a bump

In the next experiment, Fig. 6.4, we tested the LogMap framework on a plane with a bump. This example is also mentioned in (Schmidt et al., 2006) and their algorithm behaves differently from this algorithm, possibly because of their approximations based on developable surfaces. Again a cut locus is clearly visible.

6.5.3 A model problem

Finally in Fig. 6.5 the accuracy of the LogMap framework was tested using a model problem where we used a section of a sphere, for which exact distances and exact \( \log_p(x) \) functions are known. The accuracy was measured using an \( L_1 \) error measure and the results are shown in Fig. 6.6.
Figure 6.2: The Stanford bunny parameterized using polar and rectangular coordinates from the LogMap. **Top-Left:** Geodesic distance measured from a point \( p \) on the back. **Top-Right:** The angle of the LogMap coordinates has been mapped to a repeated cyclic color map and the interpretation is that sectors correspond to geodesics from \( p \). **Bottom-Left:** The LogMap coordinates expressed in the ON coordinate system, a double cyclic color map is used to display \( u \) and \( v \) texture coordinates. **Bottom-Right:** The LogMap coordinates has been used to perform local texture mapping.
6.6 Conclusions and future work

In this chapter we have introduced a new tool to the computer graphics community that is able to calculate Riemannian Normal Coordinates for triangular meshes. It uses the LogMap method in combination with an iterative algorithm for accurate estimates of geodesic distances.

We are able to show that the LogMap method on triangular meshes is accurate and has second order convergence when exact geodesic distances are known. In combination with the distance estimation used in this chapter we obtain first order convergence. Since LogMap is based on an exact mathematical formula for the log map, we conclude that it will work even for surfaces that are non-developable. In particular we note a sharp “cut locus” in the mapping at points that are antipodal to the point p.

Since the LogMap framework can be used in combination with any accurate geodesic distance estimation procedure on surfaces or manifolds, we see the need for more research estimation of geodesic distance in various settings, including level set representations of surfaces and space-variant metric tensor fields.
6.6 Conclusions and future work

Figure 6.4: Texture mapping of a plane with a bump. **Top:** A checker pattern. **Middle:** A repeated cyclic distance pattern. **Bottom:** A repeated cyclic radial pattern.

Figure 6.5: The sphere patch used for the convergence tests in Fig. 6.6. **Top-Left:** The mesh model. **Top-Right:** A checker pattern. **Bottom-Left:** A repeated cyclic distance pattern. **Bottom-Right:** A repeated cyclic radial pattern.
Figure 6.6: A comparison of Reimers algorithm versus exact distances
The skeleton of a simple closed curve $\Gamma \in \mathbb{R}^2$ is the locus of the centers of maximally inscribed open discs (Siddiqi et al., 2002), i.e. the set of open discs that are contained in the complement of $\Gamma$ such that no larger open disc contained in the complement of $\Gamma$ exists that properly contains the disc. In this chapter we describe a technique for detecting the skeleton of a closed curve in $\mathbb{R}^2$ using the LogMap framework. Since the medial locus, which is another name for the skeleton, is closely related to the cut locus on a manifold, it is not surprising that this is possible. This technique also generalizes to skeletons in $\mathbb{R}^3$ and higher dimensional Euclidean spaces, and perhaps more importantly also to skeletons in curved manifolds.

7.1 Algorithm

The skeleton of a closed curve $\Gamma$ can be calculated by a relatively simple trick, which is depicted in Fig. 7.1. The closed curve is located in $\mathbb{R}^2$ but it could also reside in some non-flat 2-manifold $M$. The intuition is that the interior of the curve is glued together with an infinitely small disc attached at the curve, making an object with the topology of a sphere for which all points at the curve $\Gamma$ are infinitely close to each other. The LogMap framework is then applied with $\Gamma$ as an abstract single base point $p$, and the Riemannian Normal Coordinates are calculated for this “point”, which is constructed by selecting $p$ on the small disc and shrink it to zero size. In the resulting coordinate system for the interior of the closed curve, the radius corresponds to the distance to $\Gamma$ and the direction correspond to the closest point on the curve, when the curve is parameterized by an angular parameter from $0$ to $2\pi$, or equivalently by a unit vector $\hat{g}$. In the normal way as seen in the previous chapters, this coordinate system is degenerate at the cut locus of $p = \Gamma$, which happens to also be the medial locus of the simple closed curve $\Gamma$.

For the practical implementation of the algorithm, we need to define the distance
Chapter 7. Estimating skeletons from LogMap

Figure 7.1: Upper-Left: The simple closed curve $\Gamma$ and an arbitrary point $x$ inside. Upper-Right: By a change of coordinates, $\Gamma$ is mapped to the circle $\Gamma'$ which has a unit metric at the curve. An additional $\epsilon$-band is defined with a radially changing metric which is $\delta_{ij}\tau$ at the border $C'$. Metric circles indicate the size of the metric. Bottom-Left: A disc with the metric $\delta_{ij}\tau$. The base point is denoted $p$ and $y$ is a point very close to $p$. The radius, $R$, depends on the metric. Lower-Right: The two discs are connected to each other by a small band $\mu$-band between $C$ and $C'$. All compositions and transitions of the metric are smooth.

from each point $x$ inside the curve to the base point $p$. This is almost the distance from $x$ to the curve $\Gamma$. However, we also need to calculate the distance from $x$ to other points $y$ very close to $p$, since the LogMap algorithm need this information. The question is how to approximate the distance to a point close to $p$.

By looking at Fig. 7.1 we may imagine a wavefront starting at a point $y$ close to $p$. Its distance to other points, or equivalently its time of arrival, can be approximated by the following formula when it travels across the small disc, the small band before it finally reaches the curve $\Gamma$ and its interior points. In the following $x$ is a point inside the closed curve in the plane. We also assume that the geodesic will travel straight across the interface between the discs, since the $\mu$-band is very thin.

$$d(y, x) = \min_{\hat{g}} d(y, C(\hat{g})) + d(C(\hat{g}), \Gamma(\hat{g})) + d(\Gamma(\hat{g}), x)$$ (7.1)

$$\approx \min_{\hat{g}} R - \langle p\hat{g}, \hat{g} \rangle + d(\Gamma(\hat{g}), x)$$ (7.2)

$$\approx \min_{\hat{g}} d(\hat{n}_g(R - y \cdot \hat{g}) + \Gamma(\hat{g}), x)$$ (7.3)

where $\hat{n}_g$ is a unit normal to $\Gamma$ pointing outwards from the closed curve, at the point on the curve pointed out by $\hat{g}$. The point $C(\hat{g})$ is a point on $C$ pointed out
by a ray from \( p \) in the direction \( \hat{g} \). These approximations should be considered in the limit when \( \tau, \mu, \epsilon \to 0 \), i.e. the patch which is glued together with the closed curve approaches zero.

The above formula may be used to calculate the distance \( d(y, x) \), either as the distance to a disturbed version of \( \Gamma \) which has been augmented by a shift in direction of the normal to \( \Gamma \), like in Eq. 7.3. Or more abstract using Eq. 7.2 where the minimization is performed over the closed curve plus a scalar contribution from \( y \). Using the formula for the LogMap framework, all that is needed is to select a few points \( y \) close to \( p \) and calculate distance maps from these points to all points \( x \) inside the closed curve. A numerical differentiation is then performed to obtain RNC.

In summary we obtain Alg. 7. Perhaps the working of the algorithm is best explained with a set of examples. In these it turns out that this framework also works for open curves in the plane.

### 7.2 Experiments

The algorithm is tested on four curves shown in Fig. 7.2, two that are closed and two that are open. The two of the additional “disturbance-curves” are also shown, corresponding to points \( y_1 = [\Delta, 0]^T \) and \( y_2 = [0, \Delta]^T \). In Fig. 7.3 the

---

**Algorithm 7** Gaussian Normal Coordinates for a closed curve \( \Gamma \).

**Require:** Let \( p = \Gamma \), a closed curve in \( \mathbb{R}^2 \) and \( x \) a point inside \( \Gamma \). The procedure \( d : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R} \) estimates Euclidean distances in \( \mathbb{R}^2 \).

**Ensure:** \( \hat{x} \in T_p M \), estimate of \( \log_p(x_i) \).

1: [Local Parameterization] Define \( \varphi : N(p) \to S \subset \mathbb{R}^2 \), a local coordinate chart around \( p \) with unit metric. Associate points \( y \) with curves \( \Gamma(\hat{g}, y) = \Gamma(\hat{g}) - (y \cdot \hat{g})\hat{n}_\hat{g} = \Gamma(\hat{g}) - \cos(\text{angle}(\hat{g}) + \text{angle}(p\hat{g})) ||p\hat{g}|| \).

2: [Distance Estimation] Estimate geodesic distances \( r(y) = d(x, y) \) from \( x \) to all closed curves \( y \in N(p) \).

3: [Function Approximation] Find an approximation \( h(z) \) to the squared geodesic distance function \( r^2(\varphi^{-1}(z)), z \in S \), using function approximation and estimates calculated in the previous step for all \( z = \varphi(y), y \in N(p) \cap X \).

4: [Gradient Calculation] Calculate estimate of \( \log_p(x) \),

\[
\hat{x} = -\frac{1}{2}(G_{ij}(z))^{-1}(\nabla h(z))^T |_{z=\varphi(p)}
\]

, where \( \nabla \) is the partial derivative, or covariant derivative, written as a row vector.
corresponding coordinate systems are shown, which breaks apart at the medial locus. In Fig. 7.4 an estimate of the medial locus has been calculated using a very simple divergence measure, not to be the optimal detection of the skeleton but mainly to illustrate that it is relatively easy to calculate from the RNC coordinate system. It turns out that this approach works well even for open curves.

7.3 Conclusion

These preliminary results show that it is possible to estimate interesting coordinate systems and skeletons for the interior of closed curves, which from the examples also seem to work for open curves. This framework should be possible to extend to other settings, for instance to the estimation of skeletons in non-flat spaces.
Figure 7.2: Simple closed and open curves in the plane. The $\sin$ and $\cos$ disturbance functions, multiplied with a large factor to make them visible, have also been plotted.
Figure 7.3: Example of estimated coordinates using LogMap. Note that the coordinate system breaks apart at points that belong to the skeleton. Colors indicate the angular argument.
Figure 7.4: An example of a skeleton estimated from coordinates derived using the LogMap framework. At the locus of the skeleton, the coordinates suddenly change, which can be detected numerically by comparing the coordinates of neighboring points.
The Riemannian exponential map, and its inverse the Riemannian logarithm map, can be used to visualize metric tensor fields. In this chapter we first derive the well-known metric sphere glyph from the geodesic equations, where the tensor field to be visualized is regarded as the metric of a manifold. These glyphs capture the appearance of the tensors relative to the coordinate system of the human observer. We then introduce two new concepts for metric tensor field visualization: geodesic spheres and geodesically warped glyphs. These additions make it possible not only to visualize tensor anisotropy, but also the curvature and change in tensor-shape in a local neighborhood. The framework is based on the \( \exp_p(v) \) and \( \log_p(q) \) maps, which can be computed by solving a second order Ordinary Differential Equation (ODE) or by manipulating the geodesic distance function. The latter can be found by solving the eikonal equation, a non-linear Partial Differential Equation (PDE), or it can be derived analytically for some manifolds. To avoid heavy calculations, we also include first and second order Taylor approximations to \( \exp \) and \( \log \). In our experiments, these are shown to be sufficiently accurate to produce glyphs that visually characterize anisotropy, curvature and shape-derivatives in smooth tensor fields.

### 8.1 Introduction

The need for tensor visualization has grown over the past twenty years along with the advancement of image analysis, computer graphics and visualization techniques. From being an abstract mathematical entity known mostly by experts in continuum mechanics and general relativity, tensors are now widely used and visualized in applied fields such as image analysis and geology. In particular, there has been an expansion over the years, from using tensors mostly in mathematical theories of the world, towards estimating tensor quantities from experimental data.

One of the most exciting areas where tensor data is derived from experiments
is the medical imaging modality called Diffusion Tensor MRI (DT-MRI). It is now becoming so central that clinical radiologists in general need to understand and visualize tensor fields representing *in vivo* water diffusion in the human brain. Fortunately, the positive definite matrices found in DT-MRI data can be visualized using ellipses (2-D) or ellipsoids (3-D), making the data understandable without knowing the details of tensor algebra. In DT-MRI, the ellipsoids are elongated along the directions of maximum water diffusion and it turns out that the shapes of them are interpretable as anatomical properties of the tissue being studied. In the human brain for instance, they are elongated in the directions of nerve fiber bundles in white matter, because water diffusion is restricted in the directions perpendicular to the fibers. In the ventricles on the other hand, where the water molecules in the cerebrospinal fluid (CSF) diffuse freely in all three directions, the ellipsoids are large and spherical. These properties of ellipsoid glyphs make DT-MRI datasets easier to comprehend for a medical expert.

Tensors are mathematical objects with special geometrical properties. Most of the research in tensor visualization has focused on the most commonly used low order tensors, in particular vectors (first order, 1-D arrays) and matrices (second order, 2-D arrays). In this chapter, we study the visualization of metric tensor fields in \( \mathbb{R}^n \), where each tensor is second order tensor. These can be represented by \( n \times n \) matrices, elements of \( \mathbb{R}^n \otimes \mathbb{R}^n \), which are symmetric and positive definite, i.e. they have positive eigenvalues. We call these tensor fields “metric tensor fields”, since they may be interpreted as the metric in a Riemannian manifold.

### 8.2 Related work

In 1881 the French cartographer Nicolas Auguste Tissot published ideas on using circles and ellipses to visualize the deformation of map projections. Mapping the Earth to a flat surface is not possible without introducing some kind of angular or area distortion in the process. The Tissot indicatrix, see Fig. 8.1, is a small circle or ellipse painted in a map projection. It represents the deformation of an infinitely small circle on the Earth after being deformed by the map projection. If the Tissot indicatrix is a perfect circle, and not an ellipse, then the projection is angle preserving (conformal), and if the area of Tissot indicatrices does not change across the map projection, the map projection is area preserving (authalic). A natural extension of the Tissot indicatrix is to use geodesic distances on the Earth (ellipsoid) to define the circle, in general resulting in a distorted ellipse. For this reason the geodesic sphere glyph we propose in this chapter, for the visualization of arbitrary metric tensor fields, can be seen as a generalization of the original Tissot indicatrix. In Fig. 8.2 we show how the geodesic variant of the Tissot indicatrix may be used to visualize the deformation of the metric in a projection of two mathematical surfaces, a half-sphere and a cone.

Later work in computer graphics has also described methods to visualize the dis-
8.2 Related work

Figure 8.1: The area and angle distortion of map projections visualized using Tissot indicatrices. **Left:** The Mercator projection, used in e.g. Google Maps. It is conformal. **Right:** The Equidistant Azimuthal projection. It is neither conformal nor authalic.

tortion of a projected surface, or manifold in general, from the information contained in a metric tensor field. In spot noise (van Wijk, 1991), a small image or spot, is pasted stochastically in multiple copies over a parametric surface to create different textures. The original paper on spot noise also demonstrates how anisotropic spot noise, in the 2-D texture coordinate system of a curved surface embedded in 3-D, results in isotropic patterns in object space. This is in fact a way to visualize the metric tensor of the surface. Textures have also been used to visualize vector fields. In line integral convolution (LIC) (Cabral and Leedom, 1993), vector fields are visualized by convolution (integration) of a random texture with streamlines created from the vector field. This yields a low-frequency response along the streamlines. In a method similar to spot noise and LIC, noise is filtered by anisotropic filters steered by second order tensors to visualize the tensor field, see for instance (Knutsson et al., 1983) for an early example or (Sigfridsson et al., 2002). Another example of second order tensor field visualization include the Hyper-LIC (Zheng and Pang, 2003), an extension of the LIC method where the convolution proceeds not only along a single streamline, but along a non-linear patch which is aligned with streamlines derived from both the first and second eigenvectors of the tensors. This is somewhat similar to the approach taken in this chapter, since a warped coordinate system is created which can be used for glyph warping. In (Hotz et al., 2004) an approach is presented based on a physical interpretation of the tensor field and it is also able to, in contrast to many other methods, visualize second order tensors with negative eigenvalues. Finally a procedural generation of textures from tensor fields have been investigated in (Kindlmann, 2004), where reaction-diffusion patterns are steered by the metric tensor field. This yields a pattern that seems to be composed by separate glyphs,
ellipses in 2-D, which are adaptively placed, scaled and deformed by the tensor field. For a successful implementation of this method, one has to overcome the numerical problems of simulating a highly non-linear PDE.

In the medical community, there has been a special need to extract information from tensor fields that goes beyond the visualization of local properties of the field. In “tractography”, entire tracts are visualized by performing streamline tracking along the main eigenvector field of a second order tensor field. This
procedure, called “fiber tracking”, helps radiologists to locate fiber bundles in the human brain and find out about long-range white matter fiber connectivity. Fiber tracking shares many similarities with the LIC, Hyper-LIC and Hyper-streamlines (Delmarcelle and Hesselink, 1993), but it is also a research topic in its own right since it is heavily biased by clinical needs and the quest for anatomical understanding of the human brain.

Two properties of spot noise and reaction-diffusion visualization seem to be important for the quality and perception of the tensor visualization. First, both of these methods spread the glyph-like spots in a uniform way according to the tensor field regarded as a metric. The latter of these methods not only scale but also bend the glyph-like structures according to the curvature of the tensor field. In recent work on glyph packing (Kindlmann and Westin, 2006) and anisotropic noise sampling (Feng et al., 2007), the first of these behaviors is mimicked and glyphs are placed uniformly over the field. However, the glyphs themselves are still based on the value of the tensor field in each point and do not consider curvature. In this chapter, we present glyphs that do exactly that: they bend, expand and contract according to the derivative of the tensor field. In combination with a glyph-packing procedure, this technique has the potential to mimic the two most desirable properties of the reaction-diffusion, in a framework that is numerically stable and fast to compute.

The work presented here is also related to work on texture mapping in computer graphics, in particular the decal compositing with discrete exponential maps (Schmidt et al., 2006). Decal compositing refers to the mapping of small texture maps, decals, onto surface models embedded in $\mathbb{R}^3$. It has been used mainly for artistic purposes and it is defined only for 2-D surfaces embedded in 3-D. Other methods for the calculation of exponential maps on general manifolds have also been presented. In (Sethian, 2001) fast marching is presented as a means to calculate geodesics emanating from a point, i.e. indirectly the calculation of exponential maps. In (Ying and Candès, 2006) fast methods are presented to calculate all geodesics in a manifold, starting from any point in any direction and traveling any distance. Finally in (Brun et al., 2005) and (Brun, 2006), the LogMap method is presented as a means of calculating the inverse of the Riemannian exponential map, a method which is reviewed later in this chapter.

### 8.3 Index notation

From here on, we will use index notation, which is commonly used in differential geometry to denote tensors and differentiate between covariant and contravariant indices. In order to make the interpretation accessible to a broader audience, we will not use the customary Einstein summation convention, meaning that all sums will be written out explicitly instead. In index notation a (contravariant) vector is identified with its coordinates, meaning that a vector $\mathbf{v}$ in Euclidean space $\mathbb{R}^n$ is
written using its coordinates $v^i$ in some basis,

$$v = v^i = \sum_{i=1}^{n} v^i b_i.$$  \hfill (8.1)\\

Note in particular that the basis vectors have been dropped and are assumed implicitly in the short form $v^i$. The index variable $i$ is an integer in the range 1 \ldots n and it is type set in superscript to indicate that this index, and this vector, is contravariant. To further increase readability we will also write equations in ordinary linear algebra notation when possible, i.e. bold face lower case letters for both contravariant and covariant vectors ($v, x, \ldots$) and upper case bold letters for matrices ($A, G, \ldots$). In some expressions we use $\dot{x}^i$ and $\ddot{x}^i$ to denote first- and second order time derivatives.

In addition to vectors, we will consider higher order tensors in this chapter, in particular the metric tensor. The metric tensor is a mathematical object which defines the scalar product between (contravariant) vectors, which in turn can be used to measure important properties in a space such as lengths, angles, area and so on. In vector algebra the scalar product is often implicitly defined simply by

$$\langle v, u \rangle = v^T u = \sum_{i=1}^{n} v^i u^i$$ \hfill (8.2)\\

but in general any symmetric positive definite $n \times n$-matrix $G$ can be used to define a metric,

$$\langle v, u \rangle_G = v^T G u = \sum_{i=1}^{n} \sum_{j=1}^{n} v^i g_{ij} u^j.$$ \hfill (8.3)\\

The latter also introduces the commonly used tensor notation for the metric, i.e. lowercase with indices written in subscript $g_{ij}$. In index notation, upper- and lower case letters have less meaning and to comply with standard notation in both linear algebra and differential geometry, we will denote the metric by either $g_{ij}$ or $G$. Subscript indices indicate that the metric is a covariant tensor. In tensor algebra it is natural to pair contravariant indices with covariant ditto, so the previous expression in Eq. 8.2 for a scalar product is somewhat odd. Instead, it is better to write out the metric explicitly,

$$\langle v, u \rangle = v^T u = \sum_{i=1}^{n} v^i \delta_{ij} u^j,$$ \hfill (8.4)\\

where $\delta_{ij}$ is the Kronecker delta symbol, defined by $\delta_{ij} = 1$ for $i = j$ and 0 elsewhere. It can be regarded as the unit-metric. Now the number of contravariant (upper) and covariant (lower) indices match, meaning that the result of the calculation is a scalar (no index).
In summary, the index notation is a handy way to denote vectors and matrices, which easily extends to higher dimensions by adding more indices. At a first glance, the common notation for vectors and matrices may seem more intuitive, but three things are easier to do in index notation. First, index notation extends naturally to higher order tensors, i.e. objects with three or more indices. Secondly, index notation can differentiate between covariance and contravariance by the use of upper- and lower indices. It should also be noted that index notation is particularly useful when used in combination with the Einstein summation convention, meaning that the summation symbol $\sum_{i=1}^{n}$ is omitted from all expressions and instead it is assumed that indices $i, j$ etc appearing more than one time in an expression is summed over, from $1 \ldots n$. In this notation the above scalar product is simply

$$\langle v, u \rangle_g = v^i g_{ij} u^j = g_{ij} v^i u^j = g_{ij} u^j v^i. \quad (8.5)$$

From the example it is also easy to see another advantage with the index notation, namely that the ordering of the tensors is irrelevant, in contrast to matrix and vector notation.

### 8.4 The metric and metric spheres

We will now take a closer look at the metric or metric tensor, and see how we can visualize a metric. We will also introduce a particular orthonormal (ON) coordinate system that will be useful later in the chapter.

The metric is the object specifying the scalar product in a particular point on a manifold in differential geometry. It encodes how to measure lengths, angles and area in a particular point on the manifold by specifying the scalar product between tangent vectors in this particular point. A natural way to visualize the metric is to visualize a “unit sphere”, i.e. a sphere with radius equal to 1. By “natural” we do not necessarily mean the most suitable way to visualize a metric from a human perception point of view, but rather a straightforward way to visualize the metric using simple mathematics. In Euclidean space the unit sphere is the set of points, $x \in \mathbb{R}^n$, satisfying $||x|| = \sqrt{\langle x, x \rangle} = 1$. In tensor notation and with an arbitrary metric $g_{ij}$ this translates to

$$\sum_{i=1}^{n} \sum_{j=1}^{n} g_{ij} x^i x^j = 1. \quad (8.6)$$

While the metric $g_{ij} = G$ may be interpreted as a symmetric positive definite matrix, it can be spectrally decomposed,

$$G = U \Lambda U^*, \quad (8.7)$$

where $U$ is a unitary matrix, $UU^* = I$, and $\Lambda$ is a diagonal matrix with the eigenvalues of $G$ ordered in descending order, $\Lambda_{ii} = \lambda_i$. The eigenvectors to $G$, found
in the columns of $U$, form an ON basis in $\mathbb{R}^n$ for both the standard metric $\delta_{ij}$ and in the arbitrary metric $g_{ij}$. For instance, in $\mathbb{R}^2$ the first eigenvector, corresponding to the eigenvalue $\lambda_1$, point along the major axis and the last eigenvector, corresponding to $\lambda_2$, point along the minor axis of the ellipse-shaped geodesic ball. In the general case, $\mathbb{R}^n$, the metric sphere will be a hyper-ellipsoid. Using this knowledge we may design a special coordinate system, which is aligned with the axes of the hyper-ellipsoid. If $U = (e_1, e_2, \ldots, e_n)$ and coordinates are denoted by $c^i$, a vector $v \in \mathbb{R}^n$ is decomposed by

$$v = v^i = \frac{1}{\sqrt{\lambda_1}} e_1 c^1 + \frac{1}{\sqrt{\lambda_2}} e_2 c^2 + \ldots + \frac{1}{\sqrt{\lambda_n}} e_n c^n. \quad (8.8)$$

![Figure 8.3: Coordinate basis vectors in $\mathbb{R}^2$ derived for some metric $g_{ij}$. This coordinate basis is ON in $g_{ij}$.](image)

This coordinate system has many advantages, in $\mathbb{R}^2$ for instance we may now easily parameterize the surface of the metric sphere by painting an isotropic sphere in the $c^i$ coordinates, $c^1 = \cos(t)$ and $c^2 = \sin(t)$, $0 \leq t < 2\pi$. An alternative approach to visualize the metric, and emphasize the direction on the eigenvectors, is to paint a unit box, $c^i : \max(c^1, c^2) = 1$. In fact, we may paint any tensor glyph in this coordinate system, for instance superquadratic tensor glyphs (Kindlmann, 2004) or even the “space ship” glyph in (Westin, 1994).

We call the map from this coordinate system to the vector space $E$, $E : \mathbb{R}^n \rightarrow V$. It is an isomorphism from the Euclidean space $\mathbb{R}^n$ (and the unit metric) to a new vector space $V$ equipped with the metric $G = g_{ij}$. Of many such isomorphisms, it has the special property that it is aligned with the axes of the hyper-ellipsoid describing $g_{ij}$ in $V$, in a particular basis.

### 8.5 The geodesic equation and geodesic spheres

In applications where metric tensor fields are visualized, the metric is not constant but changes from point to point. A natural theory for space-variant metrics is the non-Euclidean geometry found in a Riemannian manifold, which has already been pointed out by a number of authors, see for instance (O’Donnell et al., 2002). In Riemannian geometry the distance between two points in space is defined by the length of the shortest curve between them, where the length of this curve is
obtained from the integral over the tangent vectors to a curve, measured using a space-variant metric $g_{ij}(x)$,

$$d(a, b) = \min_{\gamma: \gamma(0) = a, \gamma(1) = b} \int_0^1 \sqrt{\dot{\gamma}(t)^i g_{ij}(\gamma(t)) \dot{\gamma}(t)^j} \, dt. \quad (8.9)$$

Similar to the case of a constant metric, we may now define geodesic spheres in this Riemannian manifold. For a sphere centered in a point $p$ in the manifold, the following relation hold for points $x$ in the geodesic sphere,

$$d(p, x) = 1. \quad (8.10)$$

The problem with this metric, from an application point of view, is that the space-variant metric makes it more difficult to evaluate the distance between two different points since the minimization is performed over an infinite set of curves $\gamma$.

One way to approach this problem is to derive a parametric function for points on the sphere, without measuring distances explicitly. Using the geodesic equation, geodesics emanating from a point $p$ starting off in a specific direction and traveling a specific distance (in this case 1) may be generated. These solutions correspond to paths of free particles moving in the manifold, without any forces acting on them, and in this sense they generalize the notion of straight lines in Euclidean geometry. Without going into details, geodesics can be described and calculated using the geodesic equation. It is a second order ODE, which expresses that the second derivative of the position, i.e. the acceleration, is zero. Because of the space variant metric, a special term involving the Christoffel symbol needs to be included,

$$\frac{d^2 x^i}{dt^2} + \sum_{j=1}^{n} \sum_{k=1}^{n} \Gamma_{jk}^i \frac{dx^j}{dt} \frac{dx^k}{dt} = 0, \quad (8.11)$$

where $1 \leq i, j, k \leq n$. $\Gamma_{jk}^i$ is the Christoffel symbol. It is not a tensor in a strict sense, it does not transform as a tensor when the coordinate system is changed, but it benefits greatly from the index notation since it has three indices. It is derived from the metric tensor,

$$\Gamma_{jk}^i = \frac{1}{2} \sum_{m=1}^{n} g^{im} \left( \frac{\partial g_{mj}}{\partial x^k} + \frac{\partial g_{mk}}{\partial x^j} - \frac{\partial g_{jk}}{\partial x^m} \right), \quad (8.12)$$

where $g^{ij}$ is the inverse of the metric $g_{ij}$, i.e. $g^{ij} = G^{-1}$. A geodesic starting at $\gamma(0) = p$, where $p$ is a point on the manifold, with a velocity $\dot{\gamma}(0) = v^i$ will have a geodesic length $||v||$ at $t = 1$ and thus $d(p, \gamma(1)) = ||v||$. In this way, by following geodesics starting at $p$ with different unit speed tangent vectors, we obtain a polar representation of a geodesic sphere. We will return to how this is solved in practice in a later section dealing specifically with the implementation of this.
8.6 The exponential map and Riemannian normal coordinates

With the introduction of geodesic distance and geodesics, we now have a way to paint geodesic spheres to visualize some of the characteristics of a space-variant metric tensor field. However, we have not yet introduced a coordinate system similar to the coordinates $c^i$ introduced for a constant metric. A first step towards the introduction of such a coordinate system is to define the Riemannian exponential map, known from differential geometry.

Let $T_pM$ denote the tangent space to a manifold $M$ at a point $p \in M$. In the case of our space-variant metric, this is simply the space of all tangent vectors of curves through a point $p$, which is a vector space. In particular, this is the space of all possible tangent vectors to geodesics emanating from $p$. The map $\exp_p : T_pM \rightarrow M$ is defined by $\exp_p(v^i) = \gamma(1)$, where $\gamma$ is the geodesic for which $\gamma(0) = p$ and $\dot{\gamma}(0) = v^i$. It is appropriate to use a 'shooting' analogy here, $\exp_p(v^i)$ is where a particle ends up after one time unit, if it is shot from a point $p$ with velocity $v^i$.

The introduction of the exponential map can be done without any reference to coordinates in a specific basis, it is simply a map from vectors $v^i$ seen as geometric objects in the tangent vector space of a point $p$, $T_pM$, to other points in the manifold. By choosing an ON coordinate system for $T_pM$, we obtain what is called Riemannian Normal Coordinates, Geodesic Normal Coordinates or Normal Coordinates for short. This ON basis can be seen as an isomorphism $E : \mathbb{R}^n \rightarrow T_pM$. Joining it with the exponential map, we have a map from $\mathbb{R}^n \rightarrow M$, and the inverse of this map gives us the coordinate of a point $q$ on the manifold by $\varphi(q) = E^{-1} \exp_p^{-1}(q)$, which is a well-defined inverse in a neighborhood $U$ around $p$. We will soon take a closer look at the inverse of the exponential map and call it $\log_p$.

![Figure 8.4: A schematic view of the $\exp_p$ and $\log_p$.](image)
Before we actually use the geodesic equation to paint glyphs, we will briefly touch upon how to solve it, both accurately using ODE solvers and approximately using a Taylor approximation. Like any second- or higher order ODE, it can be reformulated as a system of first order ODEs, $\frac{\partial s}{\partial t} = f(s, t)$, for a vector valued state $s$. The two variables $x^i$ and $\dot{x}^i$ evolve in time according to

$$\begin{bmatrix}
\frac{\partial x^i}{\partial t} \\
\frac{\partial \dot{x}^i}{\partial t}
\end{bmatrix} = \begin{bmatrix}
\dot{x}^i \\
- \sum_n^{j=1} \sum_n^{k=1} \Gamma^i_{jk} \dot{x}^j \dot{x}^k
\end{bmatrix},$$

(8.13)

where the $\Gamma^i_{jk}$ is spatially varying depending on $x^i$ and where the right hand side is independent of $t$. Given that initial conditions are known, e.g. $x(0) = p$ and $\dot{x}(0) = v^i$, this system of ODEs has a unique solution according to the Picard-Lindelöf theorem. While the Christoffel symbol might be difficult to comprehend at first, it is worth noting that the contribution by $\Gamma^i_{jk}$ is symmetric with respect to a flip of sign in $\dot{x}^i$. Implementation of a numerical solution to this ODE in e.g. MATLAB is straightforward using standard ODE solvers. The only reservation is that even a third order tensor-like object, like the Christoffel symbol, generates a notation which is quite involved when implemented in a vector- and matrix oriented language like MATLAB. It is also important to use a proper interpolation scheme in the calculation of derivatives of $g_{ij}$, if the tensor field is known only from samples. We used bilinear interpolation. To ensure positive definiteness we performed the interpolation in the Log-Euclidean domain (Arsigny et al., 2006b).

For many applications in computer graphics, speed and ease of implementation is an issue. For this reason we will also derive Taylor approximations of the exponential map. Directly from the geodesic equation, we have the second order derivative of our geodesic curve. Given the initial value of the position and derivative, $x(0)$ and $\dot{x}(0)$, we have everything needed in order to make a second order Taylor approximation of a geodesic, valid for small values of $t$:

$$\ddot{x}^i(t) = x^i(0) + \dot{x}^i(0)t - \frac{1}{2} \sum_n^{j=1} \sum_n^{k=1} \Gamma^i_{jk} \dot{x}^j \dot{x}^k,$$

(8.14)

which for $t = 1$ yields for a coordinate system in which $p^i = 0$,

$$\exp_p(v^i) = 0 + v^i - \frac{1}{2} \sum_n^{j=1} \sum_n^{k=1} \Gamma^i_{jk} v^j v^k + O(||v^i||^3) = q^i.$$

(8.15)

This approximation will only be valid around a small neighborhood to $p$. As of today, it is not entirely clear how good this approximation is and more research is needed, to find bounds on the approximation error and perhaps also derive higher order Taylor approximations for geodesics. As will be shown in the experimental section, this approximation is however good enough to be useful.
8.8 Geodesic spheres and warped coordinate systems

Using the formulas derived above, in particular the one derived for $\exp_p$, we are able to explicitly map unit vectors in $T_pM$ to coordinates on the manifold and thereby paint unit spheres. By choosing the special coordinate system derived above, $c^i$, in combination with these formulas, we may also navigate on the manifold using a Riemannian normal coordinate system that is aligned with the major and minor axes of the ellipse representing the unit circle. This allows us to map not only spheres, but in fact any glyph that is naturally defined in the ellipse- or ellipsoid aligned coordinate system. In this chapter we will demonstrate this by mapping the aligned unit box by using Riemannian normal coordinates. This will result in a box glyph with approximately unit length sides, which has its major axis along the main eigenvector of the local metric, but on a larger scale has its shape deformed according geodesics emanating from its center point.

8.9 The logarithmic map

The function $\log_p(q)$ is a function which maps points $q$ on the manifold to the tangent space in $p$, $T_pM$, and it is the inverse of $\exp_p(v^i)$. While the exponential function is fairly easy to calculate numerically by solving a second order ODE, the estimation of the $\log_p(q)$ mapping has attracted less attention in the literature, perhaps by the infeasibility of fast and accurate solutions. From the Taylor approximation in Eq. 8.15 it is however straightforward to derive the second order Taylor approximation for this inverse,

$$
\log_p(q^i) = 0 + q^i + \frac{1}{2} \sum_{j=1}^{n} \sum_{k=1}^{n} \Gamma_{jk}^{i} q^{j} q^{k} + O(||q^i||^3).
$$

(8.16)

In our experience this approximation is less stable than the Taylor approximation of $\exp_p(v^i)$ in Eq. 8.15, i.e. it is only valid in a small neighborhood around $p$, and for this reason we have not used the second order Taylor approximation of this mapping in our experiments.

A recently proposed method to calculate the $\log_p(q)$ map is the LogMap method (Brun, 2006; Brun et al., 2005). One way to explain this method is to study how the intrinsic mean is computed (Karcher, 1977; Fletcher et al., 2004). Let $\{x_i\}$ be $N$ data points in a manifold $M$ and seek the minimizer to the function

$$
f(p) = \frac{1}{2N} \sum_{i=1}^{N} d^2(p, x_i),
$$

(8.17)

where $d^2(p, x_i)$ is the squared geodesic distance between points $p$ and $x_i$. It is then shown in (Karcher, 1977) that, under appropriate assumptions of convexity,
the gradient of $f$ is
\[ \nabla f(p) = -g_{st} \frac{1}{N} \sum_{i=1}^{N} \log_{p} x_i. \] (8.18)

Setting $N = 1$ and $x_1 = x$ gives the following formula for $\log_{p}$,
\[ \log_{p}(x) = -g^{st} \frac{1}{2} \nabla y d^2(y, x) \bigg|_{y=p}. \] (8.19)

The metric $g_{st}$ and inverse metric $g^{st} = (g^{-1})^{st}$ have been added here to handle the general case, but choosing an ON-basis for $T_p M$ yield $g_{st} = g^{st} = \delta_{st}$ and allow us to identify co- and contravariant vectors. With the formula above, estimating $\log_{p}(q)$ becomes a matter of estimating geodesic distances on $M$. If distances $d(x, y)$ are known for all $x \in N(p)$, where $N(p)$ is some small neighborhood of $p$, and for all $y \in M$, then the gradient of the squared distance function can be easily estimated numerically by fitting a second order polynomial which is then differentiated analytically. Distance functions in turn can be estimated numerically for manifolds by solving the eikonal equation, usually by using a level-set, fast marching or even Dijkstra formalism. In some special cases (the sphere, the cone, the Poisson disk model of the hyperbolic plane, ...) the distance function can also be derived analytically. In this chapter we focus mainly on the $\exp_{p}(v^i)$ map, since it is the most convenient mapping to use if one has a glyph that is described by a set of connected vertices. We note however that if the glyph is given by a texture, the LogMap method might be convenient since it yields a mapping from points $q$ on the manifold directly to texture coordinates $v^i$. It also has the computational advantage that it calculates the mapping for all points in the manifold in one step, given only a few global distance functions from points around $p$. This property makes the LogMap method more useful when many points are to be mapped, since the ODE solution of the exponential map then requires that a large set of separate ODEs are solved.

### 8.10 Experiments

In this section we describe some experiments performed on a simulated synthetic 2-dimensional DT-MRI dataset, where noise and partial volume effects have been introduced using realistic parameter settings representative for common clinical protocols. This dataset consists of a 2-D tensor field with $2 \times 2$ symmetric positive definite tensors. We have chosen a 2-D dataset because it demonstrates several features of glyph warping and yet it is easy to visualize in print. It is however important to note that glyph warping using exponential maps is not restricted to 2-D, but works in any dimensions. In Fig. 8.5 we show a close up of the tensor field displayed using three variants of sphere-glyphs. The first variant is the metric
sphere, which may be seen as a first order approximation to the geodesic equations. The second and third image shows the second order approximation and the numerically derived solution to the geodesic ODE. In Fig. 8.6 we demonstrate the

![Figure 8.5: Left: In a first order approximation of the geodesic normal coordinates, the unit sphere is equivalent to the well-known metric ellipsoid. Middle: In a second order approximation of geodesic normal coordinates, the unit sphere might be bent. Right: When the ODE in Eq. 8.13 solved numerically, the most accurate estimates of geodesic normal coordinates are obtained. Despite the visible deformation of the Riemannian normal coordinate system attached to the center point, the geodesic sphere glyph is similar to the metric sphere in all three pictures. For this reason, geodesic spheres are not the best choice to display curvature from a human perception point of view. The box glyph overcomes some of these limitations.](image)

effect on a global scale, once again we use the sphere-glyph. The difference is subtler now, but experts in tensor image processing still agree that the two rightmost images have a softer appearance. In a third experiment, see Fig. 8.7, we

![Figure 8.6: Left: Metric sphere glyphs painted in a first order approximation of the geodesic normal coordinate system, equivalent to metric ellipsoids. Middle: Metric sphere glyphs painted in a second order approximation of geodesic normal coordinates, equivalent to metric ellipsoids. Right: Glyphs computed by solving Eq. 8.13 numerically.](image)

once again tried the three variants of glyph warping, but this time we used the box glyph instead. Here the differences are more obvious. We note that both curvature and changes in tensor shape may be seen in the two rightmost visualizations.
Again there is little difference between the second order Taylor approximation and the numerical ODE solution. Compared to the sphere-glyph, the box contains straight lines, which is the main reason why it is easier to see the effect of the non-linear mapping. In a fourth experiment, see Fig. 8.8, we tried glyph-warping on somewhat more exotic glyphs. In the image to the left we have used texture maps of soda cans as tensor glyphs. In the next image we used a creation inspired by superquadratics. Finally in the third image we have used glyph-warping on anisotropy adaptive superquadratics as defined in (Kindlmann, 2004), where isotropic glyphs have been assigned round glyphs and anisotropic glyphs have a more box-shaped appearance.

Figure 8.7: **Left:** Tensor box glyphs painted using a first order approximation of geodesic normal coordinates. **Middle:** Tensor box glyphs painted using a second order approximation of the box glyph. Note that glyphs are not only bent, they also vary in thickness that gives information that is difficult to see when painting geodesic spheres. **Right:** Glyphs computed by solving Eq. 8.13 numerically.

Figure 8.8: Warping various glyphs. **Left:** A soda can glyph. **Middle:** A superquadratic glyph. **Right:** Anisotropy adaptive superquadratic glyphs.
8.11 Conclusion

We have presented a framework for visualization of metric tensor fields in manifolds based on the Riemannian exponential map and its inverse the Riemannian logarithm map. It extends some of the previous methods for painting glyphs based on tensor eigen-decomposition and metric spheres.

Different from other proposed visualizations of tensor fields using glyphs, this glyph is not strictly a local function of the tensor field in a point, but rather the result from integration around this point in the manifold. The proposed method for warping glyphs works not only in $\mathbb{R}^2$, seen in the experiments, but also easily generalize to $\mathbb{R}^3$. By changing the glyph or modifying the tensor field, e.g. by exponentiation of the tensors, we obtain visualizations emphasizing different characteristics in the tensor field. We have derived this glyph warping from derivatives of the metric tensor field, without any reference to any embedding of the manifold (tensor field) being studied. Depending on the need for accuracy or speed, one may choose either numerically accurate geodesic warping by solving the ODE using e.g. the Runge-Kutta method or alternatively, choose the faster version where the bending of the glyphs is calculated using a Taylor approximation of the geodesic.

In summary the Riemannian exponential map, and its inverse the Logarithm map, provides a useful framework for warping glyphs and visualizing geodesics on a manifolds known only by a space-variant metric in $\mathbb{R}^n$. 
Natural metrics for parameterized image manifolds

In this chapter we will touch upon the problem of learning the metric for a parameterized image manifold. In image manifolds, each point on the manifold is an image. We will assume a low-dimensional parameterization of the manifold is already known and instead concentrate our search to find a natural metric for this manifold, derived from first principles and the data at hand. The merit of this search is to free our mind from manifold learning and embeddings, and focus on the fundamental question: What is the best metric?

9.1 Introduction

In recent years, methods for manifold learning have successfully revealed parameterizations of image manifolds from sets of sampled images, see for instance (Tenenbaum et al., 2000; Roweis and Saul, 2000). Ideally, these methods find a low-dimensional parameterization of the image manifold, \( \varphi : \mathbf{x}_i \in M \subset \mathbb{R}^N \rightarrow \mathbb{R}^d \). It is a common assumption that the manifold of images is isometrically embedded in \( \mathbb{R}^N \), which imply that the Euclidean metric locally approximate geodesic distances in the manifold, \( ||\mathbf{x}_i - \mathbf{x}_j|| \approx d(\mathbf{x}_i, \mathbf{x}_j) \) when \( \mathbf{x}_j \approx \mathbf{x}_j \), where \( d \) is the geodesic distance function.

On the other hand, there are situations where a parameterization of an image manifold is known, i.e. each image vector \( \mathbf{x}_i \) is assigned a coordinate \( z = \varphi(\mathbf{x}_i) \), \( \varphi : \mathbb{R}^N \rightarrow \mathbb{R}^d \), and \( \varphi \) is an isomorphism. In such situations it is natural to study the image manifold from an intrinsic point of view, without resorting to manifold learning to find a possibly better parameterization. The missing link between the parameterization and a complete Riemannian manifold is a metric. The metric, \( g_{ij} : T_zM \times T_zM \rightarrow \mathbb{R} \), which is implicitly understood as a function of the coordinates, \( g_{ij}(z) \).

The metric is a positive definite matrix encoding the inner product for vectors in the tangent space of the manifold. A framework for learning a suitable metric could be very useful. First, it has the potential to be a simpler problem than
manifold learning, because the topology and relevant parameterizations of the manifold are assumed to be known beforehand. Secondly, it is a fundamental problem to decide which metric is the best choice for a particular problem. This question also has some philosophical dimensions; how do we learn the metric of the world outside our body for instance?

From a theoretical point of view the metric is arbitrary, there is no such thing in Riemannian geometry as the best metric for a particular manifold. Nevertheless, there is a broad consensus that the world outside our body is a flat Euclidean space and people around the world agree on lengths, angles and area measurements up to a scaling factor. One explanation for this is that the modeling of the world, that we use to make predictions in our everyday life, generally benefit from an assumption that processes evolving in time, in the real world, are isotropic. When we see a fly for instance, we know that the distribution of velocities in which it flies is almost the same in all directions. If this is not enough, we could model its acceleration vector by an isotropic distribution. In Fig. 9.1 and Fig. 9.2 we show a simple example of how important time series can be for learning a metric. However, most algorithms for manifold learning have been designed for unordered point cloud data. They use the Euclidean distance in the embedding space as their local metric and one may ask why this works so well.

![Figure 9.1: Three linear transformations of a set of random samples. Imagine that each point is the 2-D position of a robot, derived from uncalibrated sensors. Which of the three linear transformations is metrically correct? In all three cases, the local sample density reveals only partial information about the metric. Even with the additional information that the robot is located in an office environment, excluding the image to the left because of the oblique corners, it is impossible to decide whether it is the middle or the right image that is scaled correctly.](image)

In the following sections we will turn to statistics and the study of scalar functions defined on the manifold. We will model the data by assuming it has been generated by an isotropic process and the question is which metric allow for the interpretation that the data has been generated by an isotropic process.
9.2 Related work

To the best of our knowledge, the study of the geometry of image manifolds is fairly limited apart from the many publications on manifold learning and its success on empirical data. Early modeling of images as samples from a manifold include the Self Organizing Maps (Kohonen, 1982) and Surface Learning (Bregler and Omohundro, 1994).

The modeling of sets of images as a manifold, or a curved surface embedded in a high-dimensional space, was definitely pushed forward by the work on Isomap (Tenenbaum et al., 2000) and Locally Linear Embedding (Roweis and Saul, 2000). In manifold learning, separate images are seen as points sampled from a manifold $M$ embedded in a high-dimensional Euclidean space $\mathbb{R}^N$, and the task is to recover an isometric embedding of this manifold into a low-dimensional Euclidean space $\mathbb{R}^n$, which ideally has the same dimensionality as the manifold, $\dim M = n$.

In manifold learning, in particular for images or image patches, the common assumption about the metric between pairs of images is that the distance between neighboring images can be approximated by the Euclidean distance in $\mathbb{R}^N$. For pairs of images that are far apart, no a priori assumptions are made about the distance. In e.g. Isomap, local distances are integrated by the Dijkstra shortest path algorithm to estimate long-range geodesic distances.

The celebrated embedding theorems by Nash give bounds on the maximum number of dimensions necessary for an embedding, but no manifold learning method has so far been proved to generate embeddings with optimally low dimensionality. For special cases positive results exist however. For intrinsically convex and developable (flat) manifolds with a border, Isomap will for instance reveal a globally isometric embedding into $\mathbb{R}^n$, where $\dim M = n$. 

Figure 9.2: Again the positions of the robot. This time we have displayed the random walk of the robot. The oval blob is the distribution of velocity vectors, $v_t = x_t - x_{t-1}$. If we add the prior knowledge that the robot moves according to an isotropic random walk, it is easy to conclude that the middle picture is correctly scaled.
In a paper on the Euclidean embedding of images (Donoho and Grimes, 2005), a set of image manifolds that are isometrically embedded in Euclidean space are investigated, to see in what situations e.g. the Isomap algorithm is expected to succeed. They find that many parametric image manifolds with connection to natural images are embedded isometrically in Euclidean space, making it probable that Isomap can find this embedding. They also find some counter examples where Isomap will not reveal the original parameterization, such as images of non-overlapping objects that are moved around and the image manifold is non-convex, and simple image manifolds that are non-flat such as expanding rectangles.

Finally, the term “metric learning” is defined in the paper (Lebanon, 2006) where it is used to define a parametric model of the metric in a manifold. This paper differ from the approach presented here in several aspects, in particular the approach presented here is more suited for low-dimensional manifolds.

9.3 A model for image manifolds

We assume a chart of the manifold is known, with a parameterization \( z = \varphi(x_i) \), \( \varphi : \mathbb{R}^N \supset M \rightarrow U \subset \mathbb{R}^d \). We now define a random field. The random variable \( X(\theta^a), \theta^a \subset U \), is defined on \( (\Omega, F, P) \). \( U \) is the index set of the field. \( \Omega \) is the sample space and \( w \in \Omega \). \( F \) are the events, the \( \sigma \)-algebra of subsets to \( \Omega \) for which there is a probability \( P(\ldots) \) defined. Finally, \( P \) is the function \( F \rightarrow \mathbb{R} \) that assigns a probability \( \in [0, 1] \) that satisfy:

1. \( P(E) \geq 0, \forall E \subseteq \Omega \)
2. \( P(\Omega) = 1 \)
3. \( P(E_1 \cap E_2 \cap \ldots) = \sum_i P(E_i) \) if all \( E_i \) and \( E_j \) are pairwise disjoint.

In our model we will define \( \Omega \) to be the image coordinates and \( w \in \Omega \) is a particular point or pixel. \( \theta \) is a parameter that indexes all images, i.e. it could be a parameter for generating the image in a parametric image model.

For a particular sample (pixel) \( w \), we assume that its value for different \( \theta^a \) is generated by

\[
X(\theta^a, w) = \tilde{X}(\theta^a, w)A(\theta^a) + B(\theta^a),
\]

where \( \tilde{X}(\theta^a, w) \) is a stationary (isotropic) random field on \( M \) in the following sense, for \( p, q, x_i, y_i \in M \):

\[
\exists p, q : \forall i : d(p, x_i) = d(q, y_i) \Rightarrow (\tilde{X}_w(x_1), \tilde{X}_w(x_2), \ldots, \tilde{X}_w(x_M)) \sim (\tilde{X}_w(y_1), \tilde{X}_w(y_2), \ldots, \tilde{X}_w(y_M)).
\]
9.3 A model for image manifolds

Figure 9.3: An illustration of the sample space (left) and index space (right). In our application we are interested in a metric in the index space. The sample $w$ may be thought of as a pixel or voxel, stochastically selected from the sample space $\Omega$, which in this particular case happens to be pixel positions.

Or for simplicity, that $\tilde{X}(\theta^a, w)$ is a weakly stationary (isotropic) field,

$$
E\{\tilde{X}(p)\} = E\{\tilde{X}(q)\} \quad \text{and} \quad \text{Corr}(\tilde{X}(p), \tilde{X}(q)) = \frac{\text{Cov}(\tilde{X}_w(p), \tilde{X}_w(q))}{\sqrt{E\{\tilde{X}_w^2(p)\}} \sqrt{E\{\tilde{X}_w^2(q)\}}} = r(d(p, q)),
$$

i.e. the correlation only depends on the geodesic distance between $p$ and $q$, which are two points in the manifold.

Given these assumptions, it is possible to derive an expression for the metric tensor $g_{ij}$ on $M$. Let

$$
h(q) = r(d(p, q)).
$$

Taylor approximation of the left hand side yields, for some coordinate system (where $q$ is a vector and $p = 0$),

$$
h(q) = h(0) + \nabla h(0)q + \frac{1}{2}q^T \Delta h(0)q + O(q^3)
$$

Recall from Eq. 8.16 in the previous chapter that $\log_p(q) = q + O(q^2)$ and thus $d^2(p, q) = \| \log_p(q) \|^2 = q^T g_{ij} q + O(q^3)$. We use this to Taylor approximate the
right hand side in Eq. 9.4,

\[ r(d(p, q)) = r(0) + r'(0)d(p, q) + \frac{1}{2} r''(0)d^2(p, q) + O(d^3(p, q)) \]  
\[ = r(0) + r'(0)d(p, q) + \frac{1}{2} r''(0)(q^T g_{ij} q) + O(q^3) \]  
\[ = r(0) + r'(0) \Delta h(0)/r''(0), \]  
\[ i.e. \text{ the metric is related to the Hessian of the correlation function by a scalar factor.} \]

By expressing the correlation function using Taylor series, it is possible to find a simple formula for the calculation of its Hessian. Without loss of generality we now assume that we have normalized the stochastic process, \( X = \tilde{X}, E\{X\} = 0 \) and \( \text{Var}\{X\} = 1 \). We also assume that we have centered our coordinate system on the point of interest, \( p = 0, h(q) = E\{Xw(0)\nabla Xw(0)\} = 0 \),

\[ h(q) = \frac{E\{(Xw(0) - m)(Xw(q) - m)\}}{\sigma^2} \]
\[ = E\{Xw(0)Xw(q)\} \]
\[ = E\{(Xw(0))^2 + Xw(0)\nabla Xw(0)q + \frac{1}{2} Xw(0)\Delta Xw(0)q^2\} + \ldots \]
\[ = 1 + E\{\frac{1}{2} Xw(0)\Delta Xw(0)q^2\} + O(q^3) \Rightarrow \]
\[ \Delta h(0) = E\{Xw(0)\Delta Xw(0)\}. \]

In the above we have used that \( E\{Xw(0)\nabla Xw(0)\} = 0 \). We know that the correlation function is flat for \( h(0) \), since it has a maximum at this point. Using this fact again we can also conclude that

\[ 0 = \nabla(E\{Xw(0)\nabla Xw(0)\}) \Rightarrow \]  
\[ 0 = E\{(\nabla Xw(0))^2\} + E\{Xw(0)\Delta Xw(0)\} \Rightarrow \]
\[ E\{Xw(0)\Delta Xw(0)\} = -E\{(\nabla Xw(0))^2\} \]  
\[ \text{(9.10)} \]

From this we finally arrive at

\[ g_{ij} = -E\{(\nabla Xw(0))^2\}/r''(0). \]  
\[ \text{(9.12)} \]

This is a well-known expression to most people familiar with image analysis and the so-called structure tensor (Granlund and Knutsson, 1995; Knutsson, 1989). Different from the structure tensor however, this tensor is calculated from the expectation value over the sample space \( \Omega \), i.e. it is not a spatial average like in the case of structure tensors. We call this quantity the natural metric. It is a metric that is derived from the covariant derivative (the gradient), analogous to the example with the robot previously discussed in Fig. 9.1 and Fig. 9.2, which uses contravariant vectors (the velocity) to estimate a metric from time series data.
9.4 An experiment: Intrinsic geometry in DWI

We will now set up a fairly simple experiment to test this natural metric. In diffusion tensor MRI, described earlier, the measurement is modeled using the so-called Stejskal-Tanner equation. It relates the diffusion properties of a single voxel with a measurement. The measurement is parameterized by the so-called $b$-value and the normalized direction of the gradient, $\hat{g}$. These two parameters may be seen as one single parameter vector for the measurement. In our setup for the natural metric, this MR scanner parameter vector $b = b\hat{g}$ belongs to the index set $U$ of a stochastic process. We let the sample space $\Omega$ be the set of voxels, with different diffusion properties, from which we acquire the measurements. For any single voxel $w \in \Omega$, we may acquire measurements from a range of $b$-vectors. We have chosen $b \in [0, 2000]s/mm^2$ and all possible directions. For the design of e.g. an efficient sampling of measurements in $U$, it is interesting to know if there is a natural metric defined in $U$.

The setup was the following:

- $\Omega$, the sample space, a very large set of synthetically generated voxels with realistic diffusion properties.
- $U$, the set of all $b$: $||b|| < 2000$.
- Random fields $X_w(b)$ were generated by selecting a voxel from the sample space $\Omega$ and applying the Stejskal-Tanner equation for all $b \in U$.
- The metric was estimated by Monte Carlo evaluation of the expectation value,
  \[
  g_{ij} = E\{((\nabla X_w(b))^2\}
  = E\{((\nabla X_w(b))_i(\nabla X_w(b))_j\}.
  \]
- Using the derived metric, the $\log_p(x)$ function was then used to remap the index space $U$, i.e. the set of all $b$, for $p = 0$. Because of radial symmetry, this amounts to remapping the radial argument of $b$ according to geodesic length derived from the estimated $g_{ij}$. In Fig. 9.4 this mapping function has been plotted and normalized.
- Finally as a validation, we plotted the set of all measurements for specific voxels; both in the original coordinates of $U$ and after $U$ had been remapped according to geodesic length from the natural metric $g_{ij}$. The results are shown in Fig. 9.5 – 9.7.
Figure 9.4: The remapping of the $b$-value. The scaling of the y-axis is not so relevant and it was normalized to 1. This particular remapping of the $b$-value might not be optimal in all cases since it depends on the distribution of diffusion properties in the voxels in the sample space.
9.4 An experiment: Intrinsic geometry in DWI

Figure 9.5: Top: The measurements for different $b$ before remapping. Bottom: The measurements for different $b$ after the remapping. Not that the measurement function is smoother after the mapping, making it easier to model.

Figure 9.6: Top: The measurements for different $b$ before remapping. Bottom: The measurements for different $b$ after the remapping. Not that the measurement function is smoother after the mapping, making it easier to model.
Figure 9.7: **Top:** The measurements for different $b$ before remapping. **Bottom:** The measurements for different $b$ after the remapping. Not that the measurement function is smoother after the mapping, making it easier to model.

Figure 9.8: **Top:** The measurements for different $b$ before remapping. **Bottom:** The measurements for different $b$ after the remapping. Not that the measurement function is smoother after the mapping, making it easier to model.
9.5 Conclusions

A visual inspection of the measurements before and after remapping the radial length of $b$ according to Fig. 9.4 shows that the function has a much smoother behavior after the remapping. This suggests that there is a natural metric for $U$ in this case, which in fact is not entirely exploited by remapping geodesics along radial lines if the manifold is curved.

The results in this chapter are preliminary, but a novel view on how to find a natural metric has been described, which can be applied to various settings including finding the metric of a parameterized image manifold. In this particular case, the sample space $\Omega$ can simply be the set of pixel positions in the manifold of images.

The name “natural metric” is of course chosen with the work of Amari in mind, in particular the work on the natural gradient. If there are any connections to other work in information geometry (Amari and Nagaoka, 1993) remains to be seen, apart from the superficial similarities with the Fisher information metric, which is also the expectation of an outer product.
For data samples in $\mathbb{R}^n$, the mean is a well-known estimator. When the data set belongs to an embedded manifold $M$ in $\mathbb{R}^n$, e.g. the unit circle in $\mathbb{R}^2$, the definition of a mean can be extended and constrained to $M$ by choosing either the intrinsic Riemannian metric of the manifold or the extrinsic metric of the embedding space. A common view has been that extrinsic means are approximate solutions to the intrinsic mean problem. This chapter studies both means on the unit circle and reveal how they are related to the ML estimate of independent samples generated from a Brownian distribution. The conclusion is that on the circle, intrinsic and extrinsic means are maximum likelihood estimators in the limits of high SNR and low SNR respectively.

10.1 Introduction

The mean of a set of scalar- or vector-valued data points is a well-known quantity, often used to estimate a parameter in presence of noise. Manifold-valued data is gaining importance in applications and for this kind of data several extensions of the mean have been proposed (Pennec, 1999; Gramkow, 2001; Srivastava and Klassen, 2002). While the mean for scalar- and vector-valued data can be defined as a point in the data space minimizing the sum of squared distances to all the other points, the natural extension to manifold-valued data is to replace the metric and restrict the search to a minimizer on the manifold.

10.1.1 The intrinsic mean

The intrinsic mean for a set of $N$ data points $x_i$ in a compact manifold $M$ is defined using the Riemannian metric $d_M(x, y)$, i.e. the geodesic distance between two points $x$ and $y$ in the manifold (Pennec, 1999):

$$\bar{x}_{\text{int}} = \arg \min_{q \in M} \sum_{i=1}^{N} d_M^2(x_i, q).$$
Figure 10.1: A schematic view of how the intrinsic mean (left) and extrinsic mean (right) are calculated on $S^1$. Black dots are data points and crosses mark the means. The intrinsic mean is a point on $S^1$ minimizing the sum of squared intrinsic distances (curved arrows), while the extrinsic mean is a point on the circle minimizing the sum of squared extrinsic distances (straight arrows). The white dot is an intermediate result in the calculation of the extrinsic mean, i.e. the mean of the data points in the extrinsic space $R^2$, which is followed by an orthogonal projection back to $S^1$. This procedure is equivalent to the minimization in (10.1), which explains the popularity of the extrinsic mean (Srivastava and Klassen, 2002).

While the (set of) global minimizer(s) might be difficult to compute, one may look for local minimizers, which can be guaranteed to be unique if the distributions of points $x_i$ are enough localized in $M$ (Pennec, 1999). The intrinsic mean is often seen as the natural generalization of means to manifold-valued data. The drawback is that it is relatively complicated to compute, when implemented as a (local) minimization over a non-linear manifold. The procedure is illustrated in Fig. 10.1.

### 10.1.2 The extrinsic mean

When the manifold $M$ is embedded in a Euclidean space, $R^n$, it is sometimes faster to calculate the so-called extrinsic mean. This involves two steps: 1) Calculation of the mean of the data points seen as vectors in the Euclidean space. 2) A shortest distance projection back to the manifold. This is illustrated in Fig. 10.1 and is equivalent (Srivastava and Klassen, 2002) to solving the following minimization problem

$$
\bar{x}_{\text{ext}} = \arg \min_{q \in M} \sum_{i=1}^{N} |x_i - q|^2. \tag{10.1}
$$

It is essentially the same expression as for the intrinsic mean, except that the Riemannian metric is replaced by the Euclidean metric. Note that boldface, e.g. $q$, is used when we may interpret the point as a vector in a vector space $R^n$, while $q$ is used for a point in a general manifold $M$ and sometimes refer to its coordinate.
10.2 Modeling noise by Brownian motion

It is well-known that the mean for a set of data points in \( \mathbb{R}^n \) is also the maximum likelihood (ML) estimate of \( x \) for the model \( x_i = x + n_i \), where the noise is modeled by a Gaussian distribution, \( n_i \in \mathcal{N}(0, \sigma I) \), generating a set of independent and identically distributed (i.i.d.) data points. In \( \mathbb{R}^n \) the Gaussian distribution is also a model for Brownian motion, i.e. the resulting distribution of a random walk or diffusion process. The concept of diffusion is easy to extend to manifolds in general and for this reason we choose to model noise by a Brownian distribution. We will now start with an interpretation of the mean value as the ML estimate for a model where noise in \( \mathbb{R}^n \) is modeled using Brownian motion and then proceed to the case of Brownian noise on \( S^1 \).

10.2.1 Means as ML estimates in \( \mathbb{R}^n \)

The isotropic Gaussian distribution in \( \mathbb{R}^n \) is related to Brownian motion and the diffusion equation, which is also equivalent to the heat equation. Given a distribution \( I(p, 0) \), describing the amount of particles at position \( p \) and time \( t = 0 \), the diffusion equation states

\[
I_t(p, t) = D \Delta_p I(p, t)
\]

where \( D \) is the diffusion coefficient, \( I_t \) is the derivative of \( I \) w.r.t. time and \( \Delta_p \) is the Laplacian operator acting in the spatial domain. Since \( D \) is not important in this chapter, we let \( D = 1/4 \) for simplicity. The solution to the diffusion equation at a time \( t \) is obtained by convolution in the spatial domain,

\[
I(p, t) = \int_{\mathbb{R}^n} K(p, q, t) I(q, 0) dq.
\]

\( K(p, q, t) \) is the so-called diffusion kernel,

\[
K(p, q, t) = \frac{1}{(\pi t)^{n/2}} \exp \left[ -\frac{|p - q|^2}{t} \right].
\]

To study the behavior of a single particle moving according to a Brownian motion diffusion process, one may choose \( I(p, 0) \) to be a Dirac function \( \delta(p - x) \).
Modeling noise using a Brownian (Gaussian) distribution in $\mathbb{R}^n$ now yields the following likelihood function for a set of i.i.d. data points:

$$L(x) = P(x_1, x_2 \ldots x_N | x) = P(x_1 | x) P(x_2 | x) \ldots P(x_N | x)$$

$$= C_1 \prod_{i=1}^{N} \exp \left[ -\frac{1}{t} (x_i - x)^T (x_i - x) \right]$$

$$= C_1 \exp \left[ -\frac{1}{t} \sum_{i=1}^{N} (x_i - x)^T (x_i - x) \right]$$

$$= C_2 \exp \left[ -\frac{1}{t} N(x - x)^T (x - x) \right],$$

for some constants $C_1$ and $C_2$. From this we see that regardless of $t$, the ML estimate of $x$ is the mean $\bar{x}$. We also note that both the intrinsic and extrinsic mean in $\mathbb{R}^n$ is $\bar{x}$, since $\mathbb{R}^n$ is flat.

### 10.2.2 Intrinsic means as ML estimates in $S^1$

Given the results for $\mathbb{R}^n$ it is a reasonable approach to investigate the ML estimate of i.i.d. Brownian distributions on $M = S^1$, the unit circle. The diffusion kernel on $S^1$ can be modeled using a wrapped Gaussian distribution (Strook and Turetsky, 1997),

$$K(p, q, t) = \frac{1}{\sqrt{\pi t}} \sum_{k=-\infty}^{+\infty} \exp \left[ - \frac{(d_M(p, q) + 2\pi k)^2}{t} \right]. \quad (10.3)$$

Modeling noise by $P(x_i | x) = K(x_i, x, t)$ gives an expression for the likelihood, similar to the case for $\mathbb{R}^n$, which we seek to maximize,

$$\arg \max_{x \in M} L(x) = \arg \max_{x \in M} P(x_1, x_2 \ldots x_N | x)$$

$$= \arg \max_{x \in M} P(x_1 | x) P(x_2 | x) \ldots P(x_N | x)$$

$$= \arg \max_{x \in M} \sum_{i=1}^{N} \log(P(x_i | x)).$$

Finding the ML estimate in the general case is difficult and for this reason we first study what happens in the limit when $t \rightarrow 0^+$. Due to a formula by Varadhan (Varadhan, 1967; Strook and Turetsky, 1997), it is known that

$$\lim_{t \to 0^+} t \log(K(p, q, t)) = -\frac{d_M^2(p, q)}{2}$$
uniformly in \((p, q) \in S^1 \times S^1\). For any fix \(t > 0\) we have

\[
\arg \max_{x \in M} \log(L(x)) = \arg \max_{x \in M} t \log(L(x)),
\]

and for this reason

\[
\lim_{t \to 0^+} \arg \max_{x \in M} L(x) = \lim_{t \to 0^+} \arg \max_{x \in M} t \log(L(x)) = \arg \max_{x \in M} \sum_{i=1}^{N} -\frac{d_M^2(x, x_i)}{2}
\]

\[
= \arg \min_{x \in M} \sum_{i=1}^{N} d_M^2(x, x_i)
\]

\[
= x_{\text{int}}.
\]

This means that the above ML estimate converges to \(x_{\text{int}}\) when \(t \to 0^+\).

### 10.2.3 Extrinsic means as ML estimates in \(S^1\)

Since \(L(x)\) approached \(x_{\text{int}}\) in the limit \(t \to 0^+\), it is now interesting to also investigate the behavior when \(t \to \infty\). Instead of direct use of (10.3), Fourier series are applied to solve (10.2) to obtain the diffusion kernel on \(S^1\) (Strauss, 1992). At \(t = 0\),

\[
K(p, q, 0) = \delta(d_M(p - q)) = \frac{1}{2} A_0 + \sum_{n=1}^{\infty} (A_n \cos(np) + B_n \sin(np)),
\]

\[
A_n = \frac{1}{\pi} \cos(nq) \quad (n = 0, 1, 2, \ldots)
\]

\[
B_n = \frac{1}{\pi} \sin(nq) \quad (n = 1, 2, 3, \ldots),
\]

where \(p\) and \(q\) are either points on \(S^1\) or angles in the interval \([-\pi, \pi]\). This kernel evolves according to

\[
K(p, q, t) = \frac{1}{2} A_0 + \sum_{n=1}^{\infty} e^{-n^2 t/4} [A_n \cos(np) + B_n \sin(np)].
\]

Once again, the data is modeled by \(P(x_i|x) = K(x_i, x, t)\). We observe that

\[
P(x_i|x) = \frac{1}{2\pi} + \varepsilon [A_1 \cos(x_i) + B_1 \sin(x_i)] + O(\varepsilon^2)
\]
where $\varepsilon \to 0$ when $t \to \infty$. Thus when $t \to \infty$, the likelihood function is

$$L(x) = \prod_{i=1}^{N} P(x_i|x)$$

$$= \frac{1}{(2\pi)^N} + \frac{\varepsilon}{2\pi} \sum_{i=1}^{N} [A_1 \cos(x_i) + B_1 \sin(x_i)]$$

$$+ O(\varepsilon^2).$$

Any such likelihood function will converge towards a constant value, $L(x) \to 1/(2\pi)^N$, when $t \to \infty$. The dominant terms however, important for finding the maximum of $L(x)$, are generically $A_1$ and $B_1$ and

$$\text{arg max}_{x \in M} L(x) = \text{arg max}_{x \in M} \sum_{i=1}^{N} \cos x \cos x_i + \sin x \sin x_i$$

$$= \sum_{i=1}^{N} x_i^T x_i =$$

$$= \overline{x}/\|\overline{x}\| = \overline{x}_{\text{ext}}.$$

Strange as it might seem, searching for the maximizer of a function which converges towards a constant value, it will in fact always exist a unique maximum for every $0 < t < \infty$, and generically also a unique maximizer.

### 10.3 Experiments

To verify the results we implemented the diffusion equation on the unit circle in MATLAB and calculated the likelihood as a function of $t$. The results on a small data set $x_i$ are shown for three choices of $t$ in Fig. 10.2–10.4. We also compared the intrinsic and extrinsic means for actual data sampled from Brownian distributions with different parameter $\sigma$, corresponding to $t$, on the unit circle, see Figs. 10.5–10.8. In Monte Carlo simulations, the extrinsic and intrinsic means were used as estimators for a known parameter affected by noise. The experiment clearly shows that depending on the uncertainty of the measurement, either the intrinsic or the extrinsic mean is the best choice. For the optimization process to find the global intrinsic mean, we used brute force optimization where the unit circle was sampled regularly at a fine scale.
Figure 10.2: Top: Three samples $x_i$ have been collected on $S^1$, $-2.80$, $-2.11$ and $0.34$. For $t = 0.1$ their individual likelihood functions look like in the plot. Bottom: The total normalized likelihood function $L(x)$ peaks around $-1.52$, which is close to the intrinsic mean: $\bar{x}_{\text{int}} = (-2.80 - 2.11 + 0.34)/3 \approx -1.52$. 
Figure 10.3: Same as in Fig. 10.2, but $t = 0.5$. **Top:** Individual likelihood functions. **Bottom:** The total normalized likelihood.
10.3 Experiments

\[\theta = \pi - \frac{\pi}{2} \approx -2.11.\]

Figure 10.4: Same as in Fig. 10.2, but \( t = 1.0 \). **Top:** Individual likelihood functions. **Bottom:** The total normalized likelihood peaks around \(-2.11\), which is close to the extrinsic mean:

\[\pi_{\text{ext}} = \tan^{-1}\left(\frac{\sin(-2.80) + \sin(-2.11) + \sin(0.34))}{\cos(-2.80) + \cos(-2.11) + \cos(0.34))}\right) - \pi \approx -2.11.\]
10.4 Discussion

In this chapter, we let a Brownian distribution replace the traditional Gaussian distribution. By varying the parameter $t$ we model the variance of the noise in the i.i.d. samples (measurements) $x_i \in S^1$. The signal model is a constant manifold-valued function with the value $x \in S^1$. Both the theoretical analysis and the experiments in this chapter show that the intrinsic and extrinsic means on $S^1$ can be regarded as ML estimates in the limits of high and low SNR respectively for this particular choice of models.

A close inspection of the experiment shown in Fig. 10.2–10.4, for a wider range of $t$ than shown in the figures, revealed convergence to both the intrinsic and extrinsic mean when $t \to 0^+$ and $t \to \infty$. The only reason for not including figures of experiments with very large or small $t$ in this chapter was the difficulty in obtaining a reasonable scaling of the plots. In Fig. 10.3 we observe the possibility of several local maxima for certain choices of $t$, while Fig. 10.2 and 10.4 demonstrate the typical behavior in the limits.

The result of this chapter points towards a more balanced view of the intrinsic and extrinsic means, since they are both extreme cases for our model on $S^1$. Other researchers, see for instance (Gramkow, 2001), have regarded the intrinsic mean for e.g. rotation matrices as the “natural” mean, while the extrinsic mean has been regarded as an approximation. The question is if a more balanced view, advocated in this chapter for $S^1$, is valid for a general compact manifold $M$.

Due to the generality of Varadhan’s formula (Varadhan, 1967; Strook and Turetsky, 1997), it is in fact possible to extend the results for the ML estimate when $t \to 0^+$, from $S^1$ to any connected and compact manifold. This gives a probabilistic motivation for intrinsic means on such manifolds in general. Indirectly it also motivates the use of the squared geodesic distance, $d^2_M(x, y)$, as a building block in other estimates on manifolds, for instance estimates facilitating basic interpolation and filtering. While this chapter show the essence of the idea on $S^1$, the details for the general case will be investigated in future research.

Despite the apparent symmetry of intrinsic and extrinsic means on $S^1$ presented in the paper, extending the results for the extrinsic mean and the ML estimate when $t \to \infty$ to general manifolds will not be as easy as for the case $t \to 0^+$ hinted above. In particular, the extrinsic mean depends on how the manifold $M$ is embedded in $\mathbb{R}^n$. For “natural” embeddings of certain symmetric and compact manifolds, such as $S^n$ and $\mathbb{R}P^n$, which also include important special cases such as the sphere $S^2$ and the group of rotations in $\mathbb{R}^3$, we do expect that the ML estimate will converge towards the extrinsic mean when $t \to \infty$. Thus we expect that future research will give a probabilistic motivation, based on the Brownian model of noise, for extrinsic means on e.g. the unit spheres and rotation matrices in $\mathbb{R}^n$. 
In summary, we have revealed a more balanced view on intrinsic and extrinsic means on $S^1$, which shows the essence of an idea which we believe is useful for the understanding of a wider class of algorithms performing signal processing and estimation on manifold-valued signals and data.
Figure 10.5: A comparison between extrinsic and intrinsic means for 3 Brownian samples on the unit circle. The estimation was repeated $10^6$ times for different parameters ($\sigma$) of the Brownian distribution. The standard deviation measures the efficiency of the two means.

Figure 10.6: To see clearly the difference between extrinsic and intrinsic means, the ratio between the two standard deviations has been plotted. For low amounts of noise, the intrinsic mean is the best estimator and for large amounts of noise the extrinsic mean is the best estimator.
Figure 10.7: A comparison between extrinsic and intrinsic means for 100 Brownian samples on the unit circle. The estimation was repeated $3 \times 10^4$ times for different parameters ($\sigma$) of the Brownian distribution. This time the overall deviation is smaller because of a larger amount of samples per estimation.

Figure 10.8: To see clearly the difference between extrinsic and intrinsic means, the ratio between the two standard deviations has been plotted. The difference in relative efficiency is more pronounced when the number of samples increases.
Bayesian feature space filtering

In this chapter we present a one-pass framework for filtering vector-valued images and unordered sets of data points in an $N$-dimensional feature space. It is based on a local Bayesian framework, previously developed for scalar images, where estimates are computed using expectation values and histograms. We extended this framework to handle $N$-dimensional data. To avoid the curse of dimensionality, it uses importance sampling instead of histograms to represent probability density functions. In this novel computational framework we are able to efficiently filter both vector-valued images and data, similar to e.g. the well-known bilateral, median and mean shift filters.

11.1 Introduction

In this chapter we present a method for filtering of vector-valued images, $x(q) \in \mathbb{V} = \mathbb{R}^n$, where $\mathbb{V}$ is a feature vector space such as the RGB color space. For the purposes of this chapter, $q$ is a point in a spatial vector space, $q \in \mathbb{U} = \mathbb{R}^m$, e.g. $q \in \mathbb{R}^2$ for images. It is however easy to extend this filtering to a curved $m$-dimensional manifold, $q \in M$. We also show how a slight modification can generalize this method to be used for filtering unordered sets of data points in a feature space, $\{x_i\} \in \mathbb{V} = \mathbb{R}^n$.

The proposed method is inspired by previous work by Wrangsjö et al. (Wrangsjö et al., 2004), a local Bayesian framework for image denoising of scalar-valued images. That method was based on a computational framework involving histograms, which made it slow and nearly impossible to use for vector-valued images. In this chapter we propose the novel use of a Monte Carlo method called importance sampling to overcome this difficulty. It makes this particular kind of Bayesian filtering feasible for vector-valued images and data.
11.2 Previous work

In (Wrangsjö et al., 2004) the proposed filter is related to bilateral filters (Godtliebsen et al., 1997; Lee, 1983; Smith and Brady, 1997; Tomasi and Manduchi, 1998). Other filters operating on local neighborhoods in images with similar characteristics include mean shift filtering (Comaniciu and Meer, 2002), median filters (Borik et al., 1983), total variation filters (Rudin et al., 1992), diffusion based noise reduction (Catte et al., 1992; Perona and Malik, 1990) and steerable filters (Freeman and Adelson, 1991; Knutsson et al., 1983). Several of these filters are compared in (Mrázek et al., 2006).

11.3 The Bayesian method

The method is founded on Bayesian theory and for this reason the \(a posteriori\) probability distribution function, \(p_{S|X=x}(s)\), is important. If we let \(s\) be the true value and \(x\) be the measured value which is corrupted by noise then

\[
p_{S|X=x}(s) = \frac{p_{X|S=s}(x)p_{S}(s)}{p_{X}(x)}.
\]

In order to derive an estimate \(\hat{s}\) of the true signal \(s\) from the above formula, the conditional expectation value of \(s\) may be calculated,

\[
\hat{s} = \int_{s\in\mathbb{V}} s p_{S|X=x}(s)ds = E[S|X=x]. \quad (11.1)
\]

This is the Minimum Mean Squared Error estimate, which can be calculated if the different probability distributions are modeled appropriately.

11.3.1 Noise models

The modeling of noise, how measurements are related to the true signal value, is important. For the general case, the conditional probability \(p_{X|S=s}(x)\) need to be known and in many applications this is not a problem. For the special case of additive noise, \(X = S + N\), where \(N\) can belong to e.g. a Gaussian or super-Gaussian distribution, some simplifications can be made,

\[
p_{X|S=s}(x) = \int_{t\in\mathbb{V}} \delta(x - t - s)p_{N}(t)dt
= p_{N}(x - s).
\]

For some important special cases, in particular Rician noise which is present in Magnetic Resonance (MR) images, the additive model is however not valid unless the noise is approximated using a Gaussian distribution.

It should also be mentioned that the present method only handles cases where the measurements can be considered to be independent and identically distributed.
11.3 The Bayesian method

(i.i.d.). This makes it difficult to handle e.g. speckle noise in ultrasound images efficiently.

11.3.2 Signal models for images

Most of the power of the method proposed in (Wrangsjö et al., 2004) is embedded in the \textit{a priori} p.d.f., \( p_S(s) \), which is derived from a local neighborhood around the pixel which is to be estimated. Without knowledge of the exact distribution, a kernel (Parsen window) estimate of \( p_X(x) \) is used to model a suitable local prior:

\[
p_S(s) = C_0 \left[ \sum_i b_v(x_i - s)b_s(q_0 - q_i) \right]^\alpha \approx C_0 p_X(s)^\alpha
\]  

(11.2)

where \( b_v(\cdot) \) is the kernel used to approximate density in \( \mathbb{V} \), e.g. a Gaussian, and \( b_s(\cdot) \) is a similar spatial weight which is used to favor samples which are close to \( q_0 \), the position of the pixel to be estimated. The normalizing constant \( C \) has no effect on the estimate, but the exponent \( \alpha \geq 1 \) make the histogram sharper and a higher value of \( \alpha \) promote a harder bias towards the most probable mode in the distribution \( p_X(x) \). This local modeling is ad hoc, but has proven to work surprisingly well in practice.

11.3.3 Signal models for N-D data sets

For unordered data we need to slightly modify this approach. We propose a similar way to model the \textit{a priori} distribution for unordered data, the difference being the lack of a spatial weight.

\[
p_S(s) = C_1 \left[ \sum_i b_v(x_i - s) \right]^\alpha \approx C_2 p_X(s)^\alpha
\]  

(11.4)

11.3.4 Estimation

In the original approach for scalar images, histograms were used to estimate the \textit{a priori} density function. Since the continuous integrals could not be evaluated exactly, all integrations were performed numerically in this way. In this chapter we instead propose a solution based on \textit{importance sampling} to calculate Eq. 11.1 more efficiently.
11.4 Importance sampling

In the original approach for scalar-valued images, discretized histograms were used to estimate the \textit{a priori} density function in the numerical calculation of the estimate given by Eq. 11.1. This turned out to be infeasible for vector-valued images.

It is evident that the integral in Eq. 11.1 can be evaluated using Monte Carlo, by drawing samples \( s_i \) from \( p_{S|X}(s) \) and calculate the expectation value numerically. This corresponds to the upper left illustration in Fig. 11.1. Sampling from a distribution can however be tricky and we will now introduce the concepts \textit{proper samples} and \textit{importance sampling} which will give us some freedom.

11.4.1 Proper samples

We define the following (Andrieu et al., 2002; Iba, 2001; Isard and Blake, 1998; Liu et al., 2001). A set of weighted random samples \( \{ z_i, w_i \} \), \( z_i \in p_Z \), is called
proper with respect to a distribution \( p_X \) if for any square integrable function \( h(\cdot) \),
\[
E[w_i h(z_i)] = c E[h(x_i)] \iff \int w(y)h(y)p_Z(y)dy = c \int h(y)p_X(y)dy,
\]
for some constant \( c \). Since this should be valid for any \( h(\cdot) \), \( w(y) = c p_X(y)/p_Z(y) \), and
\[
\int c p_X(y)dy = \int w(y)p_Z(y)dy
\]
\[
c = E[w(z_i)].
\]

### 11.4.2 Importance sampling

The notion of proper samples now allow us to numerically calculate the expectation value of a distribution \( p_X \) using \( M \) samples from a trial distribution \( p_Z \),
\[
E[h(x_i)] = \frac{1}{c} E[w_i h(z_i)] 
\approx \frac{1}{\sum_{i} w_i} \sum_{i} w_i h(z_i).
\]

This is how expectation values are calculated in importance sampling. It can be used when sampling from \( p_X \) is difficult but sampling from \( p_Z \) is easy. This is the case if the trial distribution \( p_Z \) is e.g. a uniform distribution, a Gaussian or a mixture of Gaussians. For us it means that we can evaluate the integral in Eq. 11.1 by sampling from another distribution \( p_Z \), if we choose the weights \( w_i \) appropriately. For the application at hand, we choose a trial distribution that is similar to the distribution of pixel-values found in the window defined by \( b_s(\cdot) \).

In figure 11.1 some examples of proper sampling are shown. Note in particular that even though evaluation using importance sampling theoretically converge to the correct expectation value when \( M \to \infty \), an unsuitable choice of trial distribution may give very slow convergence. Generically, the weight \( w_i \) for a sample \( z_i \) should be chosen so that \( w_i = p_X(z_i)/p_Z(z_i) \). If these weights grow very large, it is an indication that convergence towards the true expectation value will be slow.

### 11.5 Implementation

The Bayesian feature space filtering method was implemented in MATLAB and tested using various choices of trial functions. Two variants were derived, one for vector-valued images and one for unordered sets of data.
11.5.1 Vector-valued images

The filter was evaluated for each pixel in the image, \( x_i \) being the values of the pixels in a neighborhood large enough to fit the spatial weight function \( b_s(q) \). In the following, \( x_0 \) is the measured value in the pixel to be estimated, located at position \( q_0 \). The function \( b_v(x) \) is an isotropic Gaussian distribution with zero mean and standard deviation \( \sigma_v \), corresponding to a kernel in the feature space used in the density estimation. In the spatial domain \( b_s(q) \) is an isotropic Gaussian weight function with standard deviation \( \sigma_s \). The noise of the pixel to be estimated, \( x_0 \), is modeled using \( p_{X|S=z}(x_0) \), which is also an isotropic Gaussian distribution with standard deviation \( \sigma_n \). The conditional expectation value of \( S \) can now be expressed using the stochastic variable \( Z \), which is distributed according to the trial distribution.

\[
\bar{s} = E[S]_{X=x_0} = \int_{s \in U} sp_{S|X=x_0}(s)ds = E[Z w(Z)]/E[w(Z)]
\]

is approximated for a finite number of samples by

\[
\hat{s} = \frac{1}{\sum_{i=1}^{M} w(z_i)} \sum_{i=1}^{M} z_i w(z_i).
\]

The weight that should be used to guarantee proper samples is

\[
w(z) = \frac{p_{S|X=x_0}(z)}{p_Z(z)} = \frac{p_{X|S=z}(x_0)p_S(z)}{p_Z(z)p_X(x_0)},
\]

where \( p_X(x_0) \) is a consequence of Bayes’ rule in the derivation above, but in practice has no effect on the estimate. The prior \( p_S(z) \) is modeled using Eq. 11.2 and the trial distribution used in the sampling is a mixture of Gaussians,

\[
p_Z(z) = \frac{1}{C_3} \sum_i b_v(x_i - z), \tag{11.6}
\]

which is fairly easy to sample from. In general the choice of trial distribution is very important when implementing importance sampling. In our experiments we found that this local estimate of \( p_X \) worked well in this particular application. Generically this distribution will contain the same modes and have the same support as the a posteriori distribution we are interested in. Ignoring all constants, the weights can be calculated,

\[
w(z) = p_{X|S=z}(x_0) \left[ \sum_i b_v(x_i - z)b_s(q_0 - q_i) \right]^{\alpha} / \sum_i b_v(x_i - z).
\]
A non-stochastic alternative would have been to use the samples \( x_i \) themselves, in the neighborhood of \( x_0 \), as samples \( z_i \) and use the estimate of \( p_X \) in the neighborhood to approximate their probability density function. We implemented this variant and it worked well, but for the experiments on images reported in this chapter we have actually used true importance sampling, with a neighborhood of \( 5 \times 5 \) pixels and 125 samples \( z_i \) from the trial distribution \( Z \) in each neighborhood.

### 11.5.2 Unordered N-D data

For an unordered set of \( N \)-dimensional data, we use the prior defined in Eq. 11.4, i.e. we regard all elements in \( \{ x_i \} \) as “neighbors” to the point \( x_0 \) to be estimated, and repeat this procedure for each choice of \( x_0 \in \{ x_i \} \). The trial distribution from Eq. 11.6 is used and the lack of spatial weighting allow us to simplify the weight function,

\[
w(z) = p_{X|S=z}(x_0) \left[ \sum_i b_v(x_i - z) \right]^{\alpha-1}.
\]

Observing that the trial distribution used here is essentially the same as the distribution of points in \( \{ x_i \} \), we use approximated importance sampling in the implementation. This means that instead of sampling from the true trial distribution, we choose \( z_i = x_i \). This deterministic procedure turned out to give very similar results to true importance sampling when the number of data points was large enough.

### 11.6 Experiments

Some experiments are included to demonstrate the proposed method.

#### 11.6.1 Scalar signals

Experiments in Fig. 11.2 shows a simple example of filtering a 1-D signal. In Fig. 11.3 the method was tried out on a scalar image. These two experiments were included mainly to illustrate the behavior of the filter and show that it is similar to the previous filter proposed in (Wrangsjö et al., 2004).

#### 11.6.2 Vector-valued signals

Next the filter was tested on 2D color images, encoded as pixels with RGB color vectors. The parameters of the filters were tuned manually and Fig. 11.4 and Fig. 11.5 show examples of both good and bad parameter settings.
Figure 11.2: Filtering a 1-D scalar signal. Parameters are shown in the figure.

Figure 11.3: Filtering a noisy 2-D scalar image with outliers. **Left-Right:** Noisy data. $[\sigma_v = 0.04, \sigma_n = 100, \sigma_s = 1.0, \alpha = 1]$, $[\sigma_v = 0.04, \sigma_n = 0.5, \sigma_s = 1.0, \alpha = 20]$, $[\sigma_v = 0.04, \sigma_n = 0.5, \sigma_s = 1.0, \alpha = 5]$. 
Figure 11.4: Filtering a noisy 2-D RGB image. **Top-Left:** Noisy data. **Top-Right:** $[\sigma_v = 0.04, \sigma_n = 100, \sigma_s = 0.8, \alpha = 2]$. **Bottom-Left:** $[\sigma_v = 0.14, \sigma_n = 0.6, \sigma_s = 2.0, \alpha = 20]$. **Bottom-Right:** $[\sigma_v = 0.04, \sigma_n = 0.2, \sigma_s = 0.8, \alpha = 6]$. 
Figure 11.5: Filtering a noisy 2-D RGB image, close up of Fig. 11.4. **Top-Left**: Noisy data. **Top-Right**: $[\sigma_v = 0.04, \sigma_n = 100, \sigma_s = 0.8, \alpha = 2]$. **Bottom-Left**: $[\sigma_v = 0.14, \sigma_n = 0.6, \sigma_s = 2.0, \alpha = 20]$. **Bottom-Right**: $[\sigma_v = 0.04, \sigma_n = 0.2, \sigma_s = 0.8, \alpha = 6]$. 
11.6.3 Unordered N-D data

The filter was then tested on unordered 2-D and 3-D data, see Fig. 11.6 and Fig. 11.7. The data points in Fig. 11.7 were derived from the RGB-values of the boat image in Fig. 11.4.
Figure 11.6: Filtering unordered 2-D data. The data is a 1-D “manifold” embedded in 2-D, corrupted by noise and outliers. The gray arrows show the how each point has moved in the resulting image. **Top-Left:** Noisy data.  **Top-Right:** $\sigma_v = 0.05$, $\sigma_n = 0.05$, $\alpha = 6$.  **Bottom-Left:** $\sigma_v = 0.15$, $\sigma_n = 0.06$, $\alpha = 1$. **Bottom-Right:** $\sigma_v = 0.1$, $\sigma_n = 0.08$, $\alpha = 20$. 
11.6 Experiments

Figure 11.7: Filtering unordered 3-D data. The data is the color values from Fig. 11.4. 
Top-Left: Noisy data. Top-Right: $\sigma_v = 0.05$, $\sigma_n = 0.05$, $\alpha = 10$. 
Bottom-Left: $\sigma_v = 0.05$, $\sigma_n = 0.1$, $\alpha = 0$. Bottom-Right: $\sigma_v = 0.05$, $\sigma_n = 0.05$, $\alpha = 20$. 
11.7 Conclusion

We have presented a novel computational framework extending the previous method proposed in (Wrangsjö et al., 2004) from scalar to vector-valued images and data. The two implementations we have presented, for images and unordered data, are examples of stochastic and deterministic variants of the framework.

While the statistical modeling used here is quite simple, it should be noted that more sophisticated Bayesian modeling could be used within the same framework, for instance to model the noise more accurately for a specific application such as X-ray imaging or Diffusion Tensor MRI (DT-MRI).

It should also be noted that the proposed method based on importance sampling could also be useful for certain cases when images are scalar-valued and the dynamic range is so large that it is difficult to create histograms with the precision needed. This could be the case in computed tomography (CT).

A drawback with the method is the large number of parameters and future research will have to address this issue. Nevertheless we have found our method easy to tune and use in practice. The wide range of parameters can also be regarded as a feature since it allows the filter to change characteristics, spanning for instance both low-pass and median-like filter solutions.
This chapter briefly describes a framework for storage of geometric tensor array data, useful for storage of regularly sampled tensor fields and regularly sampled tensor-valued functions on charts of manifolds in differential geometry. The framework, called Similar Tensor Array Core headers, abbreviated STAC, captures the essence of tensor field processing in a minimalistic set of attributes. It can be used as a “greatest common divisor” in tensor processing algorithms and guide users in applied fields such as medical image analysis, visualization and manifold learning, to store and handle tensor array data in a standardized way. The framework solves many problems for new users of tensor data, promote a mathematical and geometric view of tensors and encourage exchange of tensor data between different labs and different fields of research.

12.1 Introduction

Tensors and tensor fields are basic tools in differential geometry and physics, to describe geometric and physical quantities that remain invariant under coordinate transformations. Examples include the elasticity properties of materials, diffusion and flow in the human body and local image features in 2-D and higher dimensional images. In computer programs, tensors and tensor fields are often implemented using multi-dimensional arrays, with indices corresponding to both spatial dimensions (e.g. \(x, y, \) and \(z\)) and tensor indices (e.g. \(i, j\)). Due to the lack of support for tensor data in most programming languages, individual programmers have different conventions for storing tensor data and often the programmer has limited knowledge in tensor theory and how tensor indices are related to basis (and dual basis) vectors. For this reason, we propose a standard for storage of tensor array data.

We propose a compact file format that is able to store regularly sampled real tensor fields in arbitrary dimensions. We name this data type “tensor arrays” or “geometric tensor arrays”. The approach is minimalistic, rather than flexible and
optimized. It aims to capture the essence of tensor fields, using arrays, in a way that is compatible with mathematics, physics and computer programming. To allow for a manifold of applications, we divide our work into two parts:

- The Similar Tensor Array Core headers (STAC). This is the basic data type for storage of tensor arrays. It promotes simplicity and safety in processing, communication and storage of regularly sampled tensor field data.

- Extensions. Application specific attributes and conventions for storing tensor data, for storing diffusion tensor MRI data, structure tensor fields or other kinds of data where additional information about SI units and experimental parameters need to be stored.

This chapter focuses on the core but includes examples on how to store structure tensor fields and diffusion tensor MRI data.

In the chapter we first review the few available existing standards for storage of tensor-data, we give a brief introduction to the mathematical concept of tensors and explain the mathematical notation used in the chapter. Then we present our geometrical interpretation of array-data and explain how tensors can be stored in a consistent way. A special section is dedicated to the storage of tensor array data, both conceptually with variable names and on disk, and the framework we propose is called STAC: the Similar Tensor Array Core headers. Finally we conclude with a discussion about the proposed framework and mention a few directions for future extensions.

### 12.2 Related work

Within the medical imaging and visualization community, two existing formats should be mentioned: VTK (Schroeder et al., 2000) and NRRD (Kindlmann, 2005).

VTK, the Visualization Toolkit, is able to store 0th, 1st and 2nd order tensors in 3-D using its data format. It supports the formats: structured points, structured grid, rectilinear grid, polygonal data and unstructured grid. It does not have support for separating covariant and contravariant indices and it also lacks support for higher order tensors. VTK is very versatile in describing the geometry of the data set, going far beyond regularly spaced rectangular sampling by using so called unstructured grids where data (tensors) may be attached to arbitrary points in space or to cells. VTK has been used in many applications and writers/readers for the VTK format are available in the VTK software library.

The other format, NRRD (Nearly Raw Raster Data), has support for N-dimensional data arrays. It is a fairly complex data format and it has several features that make it suitable in for instance medical applications. It is not only a tensor file format, but also a format that is able to handle non-spatial dimensions in order to describe
12.3 Geometric arrays

An array contains a set of objects, indexed by a fixed number of integers. Throughout this chapter we will follow the convention that indices start at 1. Arrays have no explicit connection to geometry, but a natural extension is to regard the $d$ indices as coordinates for a vector space $V$ spanned by an orthonormal (ON) basis. In this way, each array element corresponds to a point or cell in space. See figure 12.1. In this geometric interpretation of an array, elements are uniformly spaced in all $d$ dimensions. If elements are regarded as space-filling, like “pixels” or “voxels”, a natural interpretation is that each cell extends 0.5 units in each dimension. This is natural for most 2-D images. In medical imaging however, so called “slice thickness” may be different than the spacing between samples. Most algorithms for image processing do not take this into account and in the framework that is to be defined, STAC, the exact nature of a sample is currently not defined. Samples may be regarded as the signal measured using Dirac functions, averages over neighborhoods of various kinds or even more advanced measurement functions.

![Figure 12.1](image-url) The geometry of a $4 \times 6$ array placed in its natural coordinate system in $\mathbb{R}^2$ along with the usual Euclidean ON basis.
Table 12.1: A table of a minimalistic scalar array format.

What is important at this time is that each sample has a position in a regularly sampled grid.

12.4 Scalar array data storage

With a geometric interpretation of arrays, we now proceed to define a data format for geometric scalar arrays. Different from tensors, scalars are not geometric objects and their representations are invariant to any change of coordinate system or basis. Table 12.1 describes a minimalistic data format for geometric scalar arrays. This simple image format specifies the scalar array data, the number of spatial dimensions, the spatial size of the array and an optional metric tensor to encode the equivalent of pixel or voxel size. It also includes optional naming of the array and the indices, to make it easy to identify the scalar array with the notation of a corresponding mathematical scalar field.

The metric tensor replaces what is commonly encoded as pixel/voxel spacing in digital imaging. Given the metric tensor, we are able to measure lengths in the geometric array and it is possible to, for instance, draw a circle. It is also possible to encode oblique sampling patterns, and facilitate anisotropic processing of scalar array data even though the coordinates of the samples are natural numbers.

While there is no notion of handedness or any other information on how to transform the geometric scalar array to the real world in which we live, the standard does not say anything about how the image should be displayed. It could be displayed rotated, upside down or even mirrored. It is however important to note that most algorithms in image processing do not need this information to process scalar image data. If an application needs this information to be stored, it is something that has to be addressed in an Extension to STAC.
12.5 Tensor array data storage

Most of the problems related to storing tensors, at least in the medical engineering community, are related to transformations of tensor fields. Somewhere in the process of image acquisition and reconstruction of tensor data, there is a change of coordinate system or a transformation of the image volume (translation, rotation and scaling) without an appropriate transformation of the tensors. Tensors which were originally aligned with the image volume and for instance the anatomy of a patient (in the case of Diffusion Tensor MRI), are suddenly pointing in the completely wrong direction, see figure 12.2. The mathematically correct way to transform tensors is usually to let the tensors transform and change coordinates in a similar way that the overall geometry change, like in figure 12.3 and figure 12.4. There are exceptions from this rule, when for instance doing stretching and rotation in registration of Diffusion Tensor MRI data, the transformation of the tensor might be different but this should be seen as an exception. This has to do with the fact that stretching a volume of isotropic tensors describing cerebrospinal fluid (CSF), present in the ventricles of the brain, should usually not result in new anisotropic tensors. On the other hand, the local structure tensor in an image volume will follow the transformations of the overall geometry and the same goes for other familiar tensor fields such as flow vector fields.

Figure 12.2: An example of transforming only the geometry without changing the tensors. Note that the relation between the tensors and the geometry changes during the transformation. Typically this is not a desirable thing.

For the tensors to have meaning, we need to express them in relation to a basis. Analogous to scalar array data, tensor array data may be stretched, rotated and pasted into the real world, such as the RAS system – as long as the tensors are treated in the same way. When the tensor array is transformed into a world space, the attached tensors will get new positions and thus their coordinates have changed. In order to find out the tensor components expressed in the basis vectors used in the world space, an appropriate transformation should be applied as described earlier in Sec. 2.2. Note that this is where the information about covariant and contravariant indices really plays a role. From an application point of view however, this transformation is only needed in order to display or process
12.6 The tensor array core

The core of the tensor array standard is an almost minimal set of parameters describing the tensor array as a computational and mathematical (geometrical) object. In particular it lacks references to the physical world, including units (e.g. V, m/s, T). Despite its minimalistic qualities, it encodes a useful and self-contained block of information that may be regarded as the essence of a regularly sampled tensor field. For many tasks in basic tensor processing, including filtering, generation of tensor streamlines and calculation of the trace, the core can serve as a “greatest common divisor” in the pipeline.

The lack of references to a physical world has implications on the visualization of tensor arrays. Given only the core data, there is in fact no way to determine whether a tensor array should be displayed to a user in a regular or mirrored fashion, the “handedness” of the data set is missing. While physics in general is invariant to translation, rotation and mirroring, certain molecules called “chiral
molecules” are known to have very different biological properties compared to their mirrored versions. Furthermore in image processing, handedness is important for instance in template matching for optical character recognition (OCR). Even though handedness is important in many applications, it has not been included in the core since it describes a relation between the geometry of the tensor array and the geometry of the physical world. Most tensor processing algorithms, e.g., various kinds of filtering and interpolation techniques, are invariant under mirroring.

The core standard is described in table 12.2. It contains both optional and required data fields. The most important choice is the dimensionality of the vector space \( V \) in which the tensor array lives, \( d = \dim(V) \). If the dimensionality is 3, the array extends in three spatial dimensions and each of the tensor indices are numbered from 1 . . . 3. If for instance the array is a \( 256 \times 256 \times 2 \)-D slice of (3-D) diffusion tensors, this can only be encoded by a \( 256 \times 256 \times 1 \) volume. The spatial indices of the array may also be regarded as the spatial coordinates describing where each tensor is located. The tensor order describe the number of tensor indices each tensor has, while the index types encode whether each tensor index is contravariant or covariant. Some redundancy has been allowed for clarity, i.e., storing tensor order and array dimensionality explicitly. Some optional parameters for storing a metric tensor (analogous to voxel size) and give natural names to the tensor object have also been added. These are denoted within square brackets.

### 12.6.1 Storing array data

A convention for storage of multi-dimensional array data in a computer memory needs to be specified. Row-order, or “lexicographic order”, has been chosen for the storage of arrays. This means that first index varies slowly and last index varies fast, when storing a multi-dimensional array in a sequential memory. This is the convention for arrays in C-languages. Here it is chosen to ensure that the tensor components are stored at nearby memory locations for a single tensor (array element). It is worth to note that MATLAB and FORTRAN uses column-major order instead and a conversion is needed when data is read and written in these programming languages. The reason for not allowing both row- and column-major order is simplicity. Neither of these two schemes are optimal in all situations, ongoing research in scientific computing is for instance investigating other orderings for storing multi-dimensional arrays, such as z-order (Morton order) (Wise et al., 2001) which is a kind of space-filling curve.

For the storage of numbers, the type and byte-order should be taken into account. The core tensor standard requires data to be stored using doubles in a big-endian byte order. This requirement has been added for simplicity and it is sufficient for a wide range of applications. Efficient storage on disk may be facilitated by using file compression standards such as zip or gzip, a solution which is may be even more efficient than storing data using e.g., 16 bit integers or floats, since file
### Variable name

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Mathematical notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>tensor_name</td>
<td>$T$</td>
</tr>
<tr>
<td>tensor_index_names</td>
<td>$s(1) s(2) \ldots s(n), 1 \leq s(p) \leq d, s(p) \in \mathbb{N}$</td>
</tr>
<tr>
<td>tensor_index_types</td>
<td>$\in { \text{contravariant}', \text{covariant}' }$</td>
</tr>
<tr>
<td>tensor_order</td>
<td>$n \in \mathbb{N}$</td>
</tr>
<tr>
<td>array_index_names</td>
<td>$c^i = [c^1, c^2, \ldots, c^d]^T, 1 \leq c^i \leq m(i)$</td>
</tr>
<tr>
<td>array_metric_tensor</td>
<td>$g_{ij} \in V^* \otimes V^*$ stored in row-order and p.d.</td>
</tr>
<tr>
<td>array_size</td>
<td>$m(1) m(2) \ldots m(d), m(i) \in \mathbb{N}$</td>
</tr>
<tr>
<td>array_dimensionality</td>
<td>$d = \dim(V) \in \mathbb{N}$</td>
</tr>
<tr>
<td>data</td>
<td>$T(j) = T(j_1, \ldots, j_{d+n}) = T(c^1, \ldots, c^d, s(1) \ldots s(n)) = T(c^1, \ldots, c^d, \ldots, s(p) \ldots s(q) \ldots$, $\text{where } j \text{ maps to a row-order of } (j_1, \ldots, j_{d+n}) \text{ and } 1 \leq j \leq d^n \prod_{i=1}^d m(i)$</td>
</tr>
</tbody>
</table>

Table 12.2: A table of the tensor array core. In the example column, a $128 \times 128 \times 32$ ($d = 3$) tensor array of second order contravariant $(2, 0)$ tensors, denoted $T^{\alpha\beta}(i, j, k)$, is defined.

Compression schemes also exploit other redundancies such as often repeated data sequences.

### 12.7 Examples

A couple of examples of file headers that describe what the STAC file headers look like in practice. These files are stored with the extension “.stach” and the data is then found in a file called “.stacd”.

```python
# SIMILAR Tensor Array Core header (STAC)
# File Format release 0.9
array_dimensionality = 3
array_size = [128, 128, 32]
array_index_names = ["r", "a", "s"]
array_metric_tensor = [1, 0, 0,
                      0, 1, 0,
                      0, 0, 9]
tensor_order = 2
tensor_index_types = ["contravariant","contravariant"]
tensor_index_names = ["alpha", "beta"]
tensor_name = "T"
description = """"A diffusion tensor volume.
All tensors are positive semi definite (PSD),
```
The metric unit corresponds to 1 millimeter. The unit of the tensor $T^{ab}$ is second$^{-1}$.

In the example below STAC is used to store a local structure tensor field.

```
# SIMILAR Tensor Array Core Header (STAC)
# File Format release 0.9
array_dimensionality = 2
array_size = [64, 64]
array_index_names = ["x", "y"]
array_metric_tensor = [1, 0,
                      0, 1]
tensor_order = 2
tensor_index_types = ["covariant", "covariant"]
tensor_index_names = ["i", "j"]
tensor_name = "S"
description = "A local structure tensor field for a 64 x 64 pixel image."
```

We have implemented a reader and a writer for these STAC files in MATLAB. Since the syntax used is a subset of the Python programming language, both examples can also be tested and parsed in any Python interpreter.

12.8 Discussion

The STAC approach described here is a minimalistic framework for the storage of tensor array data. One of the main purposes of this framework is actually to point out how little information is needed to store tensor fields that can be interpreted, visualized and processed by anyone.

A peculiarity with the STAC approach is that the stored tensor data, consisting of the tensor components, is dependent of the sampling of the tensor field. If one upsample the data and thereby increase resolution, it would be equivalent to a change of coordinates and all tensor data values have to be transformed according to the transformation rules. Another effect of this is that the metric is needed in order to calculate the eigenvalues of second order tensors, simply because there is no notion of isotropy (what is round) if one does not know how to measure lengths in the geometric array. These effects are somewhat unexpected to a novice user, but it also forces users of the standard to actually know what a tensor is and how to handle transformations, even for simple cases. And if the user knows this, the user knows everything about tensors – since tensors are very simple geometric objects.
Summary and outlook

This book has discussed various aspects of the use of manifolds in image science and visualization. It has been a fairly broad presentation with topics ranging from mean values on circles to texture mapping in computer graphics. Even though these topics may seem remote from an application point of view, it has been long known that the intrinsic mean in a manifold may be numerically calculated using the $\log_p(x)$ function and it should now also be clear to the reader that texture mapping may use the very same machinery. But of course, the broad list of topics it is also an example of the non-linear process of research into unknown territories.

13.1 Future Research

- In dimension reduction and manifold learning, current state of the art algorithms have still not converged to the best solution to learn the underlying manifold for a set of unordered samples. The work on LogMaps presented in this dissertation is a unique and canonical approach to find a coordinate system for a manifold, which we believe has a great potential. The major drawback so far has been the lack of accurate distance estimation on sampled manifolds, since Dijkstra’s algorithm used in the original LogMap paper produced fairly noisy mappings. However, the work in this dissertation on texture mapping using LogMaps clearly shows that LogMaps have a lot to offer if accurate distance estimates are given.

- The texture mapping algorithm is currently being evaluated for other methods for distance estimation, including the Fast Marching Method and alternative ways to represent surfaces in computer graphics, such as the level set approach.

- Given accurate distance estimates in arbitrary metric tensor fields, that itself is a hard problem to solve, the LogMap framework should be possible to use to perform fiber tracking in the brain. It would be a method to instantly find all the geodesic curves from a single seed point in white matter to all other
• The skeleton algorithm using LogMaps may have future applications, but at the moment it is mostly a curiosity since there are other ways to estimate skeletons in e.g. binary images. An interesting detail however is that the LogMap framework both works in arbitrary dimensions and even for curved manifolds with a border, given that accurate geodesic distance estimates are available. Secondly the LogMap framework is able to estimate a coordinate system, as well at the medial locus.

• The geodesic glyph warping has still not been tried in 3-D. It should be straightforward but it is still an open question whether it will be of benefit to the user to visualize curvature in 3-D. A next step is to combine this method with glyph packing algorithms too.

• The natural metric. This was a fairly recent addition. The fundamental problem of learning a suitable metric for a particular task will most probably continue to be important. For the natural metric here presented, the next step is to continue the investigation on more real world applications.

• Intrinsic and extrinsic means. This line of research should be possible to extent to other manifolds than the circle, such as the sphere and the group of rotations, $SO(3)$.

• The Bayesian feature space filtering originally came into existence from the need one day to filter some unordered data. One interesting line of research to pursue would be to investigate if this method could be applied to “non-local means” which deals with image patches (Buades et al., 2005).

• The proposal for a tensor standard is currently being finalized for presentation in a journal. In the long run, there is a need for a more standardized way to exchange tensor- and manifold data, but it is of course difficult to see which standard could have the potential to fit all users since tensor signal processing is a fairly broad field of research with applications ranging from medical image science to geology.

There is still a lot of work to be done in the area of differential geometry, image science and visualization. One cannot hope for a unified framework for this in the near future, but perhaps a set of tools will crystallize and become as common as PCA, structure tensors and scatter plots are today.


URL http://www.cs.utah.edu/research/techreports.shtml


port LiTH–ISY–I–1019, Computer Vision Laboratory, Linköping University, Sweden, 1989.


Pearson, K., 1901. On lines and planes of closest fit to systems of points in space. Philosophical Magazine 2, 559–572.


