Differential Methods for Multi-dimensional Visual Data Analysis

Werner Benger, René Heinzl, Dietmar Hildenbrand, Tino Weinkauf, Holger Theisel, and David Tschumperlé

Contents

1 Introduction 2100
2 Modeling Data via Fiber Bundles 2102
Differential Geometry: Manifolds, Tangential Spaces, and Vector Spaces 2103
Topology: Discretized Manifolds 2111
Ontological Scheme and Seven-Level Hierarchy 2113
3 Differential Forms and Topology 2119
Differential Forms 2119
Homology and Cohomology 2128

W. Benger (✉)
Airborne Hydromapping Software GmbH, Innsbruck, Austria

Institute for Astro- and Particle Physics, University of Innsbruck, Innsbruck, Austria

Center for Computation and Technology, Lousiana State University, Baton Rouge, LA, USA
e-mail: werner@cct.lsu.edu

R. Heinzl
Shenteq s.r.o, Bratislava, Slovak Republic
e-mail: heinzl@shenteq.com

D. Hildenbrand
University of Technology Darmstadt, Darmstadt, Germany

T. Weinkauf
Feature-Based Data Analysis for Computer Graphics and Visualization, Max Planck Institute for Informatics, Saarbrücken, Germany
e-mail: weinkauf@mpi-inf.mpg.de

Holger Theisel
Institut fur Simulation und Graphik AG Visual Computing, Magdeburg, Germany
e-mail: theisel@isg.cs.uni-magdeburg.de

David Tschumperlé
GREYC (UMR-CNRS 6072), CAEN Cedex, France
e-mail: david.tschumperle@greyc.ensicaen.fr

O. Scherzer (ed.), *Handbook of Mathematical Methods in Imaging*,
DOI 10.1007/978-1-4939-0790-8_35

Topology 2130
4 Geometric Algebra Computing 2132
Benefits of Geometric Algebra 2134
Conformal Geometric Algebra 2136
Computational Efficiency of Geometric Algebra Using Gaalop 2139
5 Feature-Based Vector Field Visualization 2141
Characteristic Curves of Vector Fields 2141
Derived Measures of Vector Fields 2143
Topology of Vector Fields 2145
6 Anisotropic Diffusion PDEs for Image Regularization and Visualization 2149
Regularization PDEs: A Review 2149
Applications 2154
7 Conclusion 2157
Cross-References 2159
References 2159

Abstract

Images in scientific visualization are the end product of data processing. Starting from higher-dimensional data sets such as scalar, vector, and tensor fields given on 2D, 3D, and 4D domains, the objective is to reduce this complexity to two-dimensional images comprehensible to the human visual system. Various mathematical fields such as in particular differential geometry, topology (theory of discretized manifolds), differential topology, linear algebra, Geometric Algebra, vector field and tensor analysis, and partial differential equations contribute to the data filtering and transformation algorithms used in scientific visualization. The application of differential methods is core to all these fields. The following chapter will provide examples from current research on the application of these mathematical domains to scientific visualization. Ultimately the use of these methods allows for a systematic approach for image generation resulting from the analysis of multidimensional datasets.

1 Introduction

Scientists need an alternative to numbers. The use of images is a technical reality nowadays and tomorrow it will be an essential requisite for knowledge. The ability of scientists to visualize calculations and complex simulations is absolutely essential to ensure the integrity of analyses, to promote scrutiny in depth and to communicate the result of such scrutiny to others… The purpose of scientific calculation is looking, not enumerating. It is estimated that 50 % of the brain's neurons are associated with vision. Visualization in a scientific calculation is aimed at putting this neurological machinery to work [56].

Since this visionary quote from an article in 1987, scientific visualization, benefiting from the affordable graphics hardware driven by the computer gaming industry, has grown rapidly. Beyond academic research interests it has become also a consumer market with practical applicability in industry and medicine. Still there are yet many gaps that are left open due to the unequal evolution velocities in different fields. Once, there is the human mind that is not able to keep up with the deluge of visual information which can be produced with modern technology. Many scientists still prefer looking at numbers instead of utilizing modern display technology. At the same time, data can be produced by modern supercomputers that is far beyond the ability of even high-end graphics engines to be processed. Data sets originating from numerical simulations of physical processes will usually be three-dimensional or four-dimensional, with images just the final result of the process of scientific visualization. In this context images are the means to analyze data set of higher dimensions.

Reducing numerical data sets to images is known as the concept of the *visualization pipeline*. In its simplest form it consists of a data source (n-dimensional), a data filter (an algorithmic operation), and a data sink (an image). Data filters need to understand the structure and meaning of the multidimensional input data and to operate efficiently on them. This involves various mathematical fields such as in particular differential geometry, topology (theory of discretized manifolds), differential topology, linear algebra, Geometric Algebra, vector field and tensor analysis, and partial differential equations. Within a scientific visualization process, all these mathematical fields will work together, with more or less weighting. We subsume this set of mathematical domains as “differential methods” in this chapter as the concept of differentiation is fundamental to their approach of data analysis. The following sections will demonstrate the application of the respective mathematical fields to visual analysis by virtue of examples of ongoing research.

In Sect. 2 we discuss the general issue of how to lay out data to model the structure of space and time, as we know it from mathematics as foundation for further operations. Frequently visualization algorithms are implemented ad hoc, given the problem, inventing the solution with highest performance. This allegedly reasonable approach comes with an unfortunate downside: incompatibility among independently developed solutions, which impacts data exchange and interfacing complementary implementations. However, when keeping a common data model in mind right from the earliest steps of conceiving some algorithm, interoperability can be achieved at no cost with same performance as solitary solutions.

Given a solid foundation for data structures, Sect. 3 demonstrates how to formulate differential operators using the concepts of chains, cochains, homology, and cohomology. Since in computer graphics and visualization we have to deal with discretized spaces, we arrive in the mathematical field of topology, as an essential descriptive tool for meshes and all nontrivial grid structures.

When considering mathematics as a language unifying computer science, we need to even more think about a common denominator within mathematics itself. Geometric Algebra is a relatively new – or, rather, rediscovered – branch of mathematics that is very promising. It is extraordinarily visually intuitive while covering the abstractions of Clifford algebra as used in quantum mechanics equally well as the formulations of curved space in general relativity. However, even independent of such physics-oriented applications, Geometric Algebra has found its merits within computer graphics itself. Section 4 will talk about the elegant usage of five-dimensional projective conformal Geometric Algebra to handle primitives in computer graphics and eventually implement the ray-tracing algorithm with a few, well-defined algebraic operations.

The general goal of visualization is to give insight into large and complex data sets. Due to the sheer size of the data sets alone, it is favorable if not necessary to automate at least parts of the analysis. A way to achieve this is by extracting features. Features can either be certain quantities derived from a data set or a mathematically well-defined, geometric object (point, line, surface, ...) with its definition and interpretation depending on the underlying application, but usually it represents important structures (e.g., vortex, stagnation point) or changes to such structures (events, bifurcations). A feature-based visualization aims at the reduction of information to guide a user to the most interesting parts of a data set. In Sect. 5 we describe some important approaches to feature-based visualization of vector fields. These include investigation of derived quantities such as vortices (section “Derived Measures of Vector Fields”) and the topology of vector fields (section “Topology of Vector Fields”). These approaches have become a standard tool for the analysis of vector fields.

Finally, in Sect. 6 we explore the capabilities of partial differential equations for the filtering and regularization of image data sets. Applications are enhancing image quality by reducing noise or similar artifacts, as well as the visualization of vector and tensor fields.

2 Modeling Data via Fiber Bundles

Purely numerical algorithms in C++ can be abstracted from concrete data structures using programming techniques such as generic programming [79]. However, generic algorithms still need to make certain assumptions about the data they operate on. The question remains what these *concepts* are that describe “data”: what properties should be expected by some algorithm from any kind of data provided for scientific visualization? Moreover, consistency among concepts shared by independent algorithms is also required to achieve *interoperability* among algorithms and eventually (independently developed) applications. While any particular problem can be addressed by some particular solution, a common concept allows to build a *framework* instead of just a collection of *tools*. Tools are what an end user needs to solve a particular problem with a known solution. However, when a problem is not yet clearly defined and a solution unknown, then a framework is required that allows exploration of various approaches and eventually adaption toward a specific direction that does not exist a priori.

The concept of how to lay out data to perform visualization operations in a common framework constitutes a *data model* for visualization. Many visualization applications are to a greater or lesser extent a collection of tools, even when bundled together within the same software library or binary. Consequently, interoperability between different applications and their corresponding file formats is hard or impossible. Only very few implementations adhere to the vision of a common data model across the various data types for visualization. The idea of a common data model is frequently undervalued or even disregarded as being impossible. However, as D. Butler said, "The proper abstractions for scientific data are known. We just have to use them" [16].

D. Butler was following the mathematical concepts of fiber bundles [16], or more specific, vector bundles [15], to model data. The IBM Data Explorer, one of the earliest visualization applications, now open source and known as "OpenDX (http://www.opendx.org)," implemented this concept successfully [76]. These ideas have been revived and expanded by [7] leading to a hierarchical data structure consisting of a noncyclic graph in seven levels. It can be seen as largely keyword-free, hierarchical version of the OpenDX model, seeking to cast the information and relationships provided in original model into a grouping structure. This data model will be reviewed in the following, together with its mathematical background. Section "Differential Geometry: Manifolds, Tangential Spaces, and Vector Spaces" will review the basic mathematical structures that are used to describe space and time. Section "Topology: Discretized Manifolds" will introduce the mathematical formulation of discretized space. Based on this background, section "Ontological Scheme and Seven-Level Hierarchy" will present a scheme that is able to cover the described mathematical structures.

Differential Geometry: Manifolds, Tangential Spaces, and Vector Spaces

Space and time in physics is modeled via the concept of a differentiable manifold. As scientific visualization deals with data given in space and time, following these concepts is reasonable. In short, a manifold is a topological space that is locally homeomorphic to $\mathbb{R}^n$. However, not all data occurring in scientific visualization are manifolds. The more general case of topological spaces will be discussed in sections "Topology: Discretized Manifolds" and "Topology."

A vector space over a field F(such as $\mathbb{R}$) is a set V together with two binary operations *vector addition* $+ : V \times V \to V$ and *scalar multiplication* $\circ : F \times V \to V$. The mathematical concept of a *vector* is defined as an element $v \in V$. A vector space is closed under the operations $+$ and $\circ$, i.e., for all elements $u, v \in V$ and all elements $\lambda \in F$ there is $u + v \in V$ and $\lambda \circ u \in V$ (vector space axioms). The vector space axioms allow computing the differences of vectors and therefore defining the derivative of a vector-valued function $v(s) : \mathbb{R} \to V$ as

$$\frac{d}{ds} v(s) := \lim_{ds \to 0} \frac{v(s + ds) - v(s)}{ds} \tag{1}$$

A manifold in general is *not* a vector space. However, a differentiable manifold M allows to define a tangential space $T_P(M)$ at each point P which has vector space properties.

Tangential Vectors

In differential geometry, a tangential vector on a manifold M is the operator $\frac{d}{ds}$ that computes the derivative along a curve $q(s) : \mathbb{R} \to M$ for an arbitrary scalar-valued function $f : M \to \mathbb{R}$:

$$\frac{d}{ds} f \bigg|_{q(s)} := \frac{df(q(s))}{ds} \tag{2}$$

Tangential vectors fulfill the vector space axioms and can therefore be expressed as linear combinations of derivatives along the n coordinate functions $x^\mu : M \to \mathbb{R}$ with $\mu = 0 \ldots n - 1$, which define a basis of the tangential space $T_{q(s)}(M)$ on the n-dimensional manifold M at each point $q(s) \in M$:

$$\frac{d}{ds} f = \sum\nolimits_{\mu=1}^{n-1} \frac{dx^\mu(q(s))}{ds} \frac{\partial}{\partial x^\mu} f =: \sum\nolimits_{\mu=1}^{n-1} \dot{q}^\mu \partial_\mu f \tag{3}$$

where $\{q\}^\mu$ are the components of the tangential vector $\frac{d}{ds}$ in the chart $\{x^\mu\}$ and $\{\partial_\mu\}$ are the basis vectors of the tangential space in this chart. In the following text the Einstein sum convention is used, which assumes implicit summation over indices occurring on the same side of an equation. Often tangential vectors are used synonymous with the term "vectors" in computer graphics when a direction vector from point A to point B is meant. A tangential vector on an n-dimensional manifold is represented by n numbers in a chart.

Covectors

The set of operations df: $T(M) \to \mathbb{R}$ that map tangential vectors $v \in T(M)$ to a scalar value $v(f)$ for any function $f : M \to \mathbb{R}$ defines another vector space which is dual to the tangential vectors. Its elements are called *covectors*:

$$< df, v >= df(v) := v(f) = v^\mu \partial_\mu f = v^\mu \frac{\partial f}{\partial x^\mu} \tag{4}$$

Covectors fulfill the vector space axioms and can be written as linear combination of covector basis functions dx^μ:

$$df =: \frac{\partial f}{\partial x^\mu} dx^\mu \tag{5}$$

whereby the dual basis vectors fulfill the duality relation

$$< dx^\nu, \partial_\mu >= \begin{cases} \mu = \nu : & 1 \\ \mu \neq \nu : & 0 \end{cases} \tag{6}$$

The space of covectors is called the cotangential space $T_P{}^*(M)$. A covector on an n-dimensional manifold is represented by n numbers in a chart, same as a tangential vector. However, covectors transform inverse to tangential vectors when changing

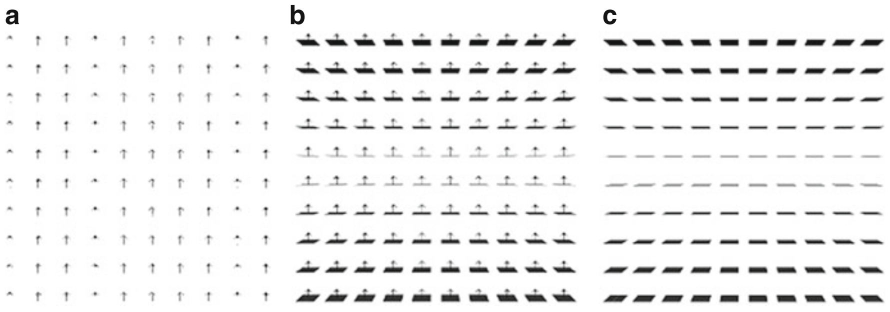


Fig. 1 The (trivial) constant vector field along the z-axis viewed as vector field ∂_z and as covector field dz. (**a**) Vector field ∂_z. (**b**) Duality relationship among ∂_z and dz. (**c**) Co-vector field dz

coordinate systems, as is directly obvious from Eq. (6) in the one-dimensional case: as $< dx^0, \partial_0 >= 1$ must be sustained under coordinate transformation, dx^0 must shrink by the same amount as ∂_0 grows when another coordinate scale is used to represent these vectors. In higher dimensions this is expressed by an inverse transformation matrix.

In Euclidean three-dimensional space, a plane is equivalently described by a "normal vector," which is orthogonal to the plane. While "normal vectors" are frequently symbolized by an arrow, similar to tangential vectors, they are not the same, rather they are dual to tangential vectors. It is more appropriate to visually symbolize them as a plane. This visual is also supported by (5), which can be interpreted as the total differential of a function f: a covector describes the change of a function f along a direction as specified by a tangential vector $\vec{v}$. A covector V can thus be visually imagined as a sequence of coplanar (locally flat) planes at distances given by the magnitude of the covector that count the number of planes which are crossed by a vector $\vec{w}$. This number is $V(w)$. For instance, for the Cartesian coordinate function x, the covector dx "measures" the "crossing rate" of a vector w in the direction along the coordinate line x; see Figs. 1 and 2. On an n-dimensional manifold a covector is correspondingly symbolized by a $(n-1)$-dimensional subspace.

Tensors

A *tensor* T_n^m of rank $n \times m$ is a multi-linear map of n vectors and m covectors to a scalar

$$T_n^m : T(M) \times \ldots T(M)_n \times T^*(M) \times \ldots T^*(M)_m \to \mathbb{R} \tag{7}$$

Tensors are elements of a vector space themselves and form the tensor algebra. They are represented relative to a coordinate system by a set of k^{n+m} numbers for a k-dimensional manifold. Tensors of rank 2 may be represented using matrix notation. Tensors of type $T_1{}^0$ are equivalent to covectors and called co-variant; in

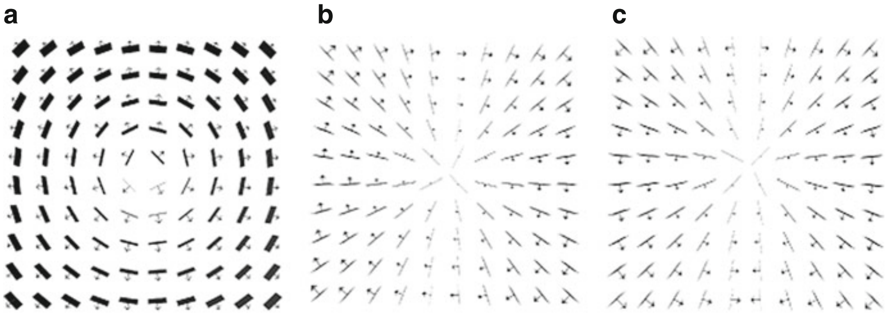


Fig. 2 The basis vector and covector fields induced by the polar coordinates $\{r, \vartheta, \phi\}$. (**a**) Radial field dr ∂_r. (**b**) Azimuthal field dϕ ∂_ϕ view of the equatorial plane (z-axis towards eye). (**c**) Altitudal field dθ ∂_θ slice along the z-axis

matrix notation (relative to a chart) they correspond to rows. Tensors of type $T_0{}^1$ are equivalent to a tangential vector and are called contra-variant, corresponding to columns in matrix notation. The duality relationship between vectors and covectors then corresponds to the matrix multiplication of a $1 \times n$ row with a $n \times 1$ column, yielding a single number

$$< a, b >=< a^\mu \partial_\mu, b_\mu dx^\mu > \equiv \; (a^0 a^1 \ldots a^n) \begin{pmatrix} b^0 \\ b^1 \\ \ldots \\ b^n \end{pmatrix} \tag{8}$$

By virtue of the duality relationship (6), the contraction of lower and upper indices is defined as the *interior product* ι of tensors, which reduces the dimensionality of the tensor:

$$\iota : T_n^m \times T_k^l \rightarrow T_{n-1}^{m-k} : u, v \mapsto \iota_u v \tag{9}$$

The interior product can be understood (visually) as a generalization of some "projection" of a tensor onto another one.

Of special importance are symmetric tensors of rank two $g \in T_2^0$ with $g : T(M) \times T(M) \rightarrow \mathbb{R} : u, v \mapsto g(u, v), g(u, v) = g(v, u)$, as they can be used to define a *metric* or *inner product* on the tangential vectors. Its inverse, defined by operating on the covectors, is called the co-metric. A metric, same as the co-metric, is represented as a symmetric $n \times n$ matrix in a chart for an n-dimensional manifold.

Given a metric tensor, one can define equivalence relationships between tangential vectors and covectors, which allow to map one into each other. These maps are called the "musical isomorphisms," $\flat$ and $\sharp$, as they raise or lower an index in the coordinate representation:

$$\flat : \quad T(M) \to T^*(M) : \quad v^\mu \partial_\mu \mapsto v^\mu g_{\mu\nu} dx^\nu \tag{10}$$

$$\sharp : \quad T^*(M) \to T(M) : \quad V_\mu dx^\mu \mapsto V_\mu g^{\mu\nu} \partial_\nu \tag{11}$$

As an example application, the "gradient" of a scalar function is given by $\nabla f = \sharp df$ using this notation. In Euclidean space, the metric is represented by the identity matrix and the components of vectors are identical to the components of covectors. As computer graphics usually is considered in Euclidean space, this justifies the usual negligence of distinction among vectors and covectors; consequently graphics software only knows about one type of vectors which is uniquely identified by its number of components. However, when dealing with coordinate transformations or curvilinear mesh types, distinguishing between tangential vectors and covectors is unavoidable. Treating them both as the same type within a computer program leads to confusions and is not safe.

Exterior Product

The *exterior product* $\wedge : V \times V \to \Lambda(V)$ is an algebraic construction generating vector space elements of higher dimensions from elements of a vector space V. The new vector space is denoted $\Lambda(V)$. It is alternating, fulfilling the property $v \wedge u = -u \wedge v \ \forall u, v \in V$ (which results in $v \wedge v = 0 \ \forall v \in V$). The exterior product defines an algebra on its elements, the exterior algebra (or Grassmann algebra). It is a sub-algebra of the tensor algebra consisting of the antisymmetric tensors. The exterior algebra is defined intrinsically by the vector space and does not require a metric. For a given $n - \textit{dimensional}$ vector space V, there can at most be nth power of an exterior product, consisting of n different basis vectors. The $(n + 1)$th power must vanish, because at least one basis vector would occur twice, and there is exactly one basis vector in $\Lambda^n(V)$.

Elements $v \in \Lambda^k(V)$ are called k-vectors, whereby two-vectors are also called bi-vectors and three-vectors tri-vectors. The number of components of a k-vector of an n-dimensional vector space is given by the binomial coefficient $\{n\}\{k\}$. For $n = 2$ there are two one-vectors and one bi-vector, for $n = 3$ there are three one-vectors, three bi-vectors, and one tri-vector. These relationships are depicted by the Pascal's triangle, with the row representing the dimensionality of the underlying base space and the column the vector type:

$$\begin{array}{ccccccccc} & & & & 1 & & & & \\ & & & 1 & & 1 & & & \\ & & 1 & & 2 & & 1 & & \\ & 1 & & 3 & & 3 & & 1 & \\ 1 & & 4 & & 6 & & 4 & & 1 \end{array} \tag{12}$$

As can be easily read off, for a four-dimensional vector space, there will be four one-vectors, six bi-vectors, four tri-vectors, and one four-vector. The n-vector of an n-dimensional vector space is also called a *pseudoscalar*, the $(n-1)$ vector a *pseudo-vector*.

Visualizing Exterior Products

An exterior algebra is defined on both the tangential vectors and covectors on a manifold. A bi-vector v formed from tangential vectors is written in chart as

$$v = v^{\mu\nu}\partial_\mu \wedge \partial_\nu \tag{13}$$

and a bi-covector U formed from covectors is written in chart as

$$U = U_{\mu\nu}dx^\mu \wedge dx^\nu \tag{14}$$

They both have $\{n\}\{2\}$ independent components, due to $v^{\mu\nu} = -v^{\nu\mu}$ and $U_{\mu\nu} = -U_{\nu\mu}$ (three components in 3D, six components in 4D). A bi-tangential vector (13) can be understood visually as an (oriented, i.e., signed) plane that is spun by the two defining tangential vectors, independently of the dimensionality of the underlying base space. A bi-covector (14) corresponds to the subspace of an n-dimensional hyperspace where a plane is "cut out." In three dimensions these visualizations overlap: both a bi-tangential vector and a covector correspond to a plane, and both a tangential vector and a bi-covector correspond to one-dimensional direction ("arrow"). In four dimensions, these visuals are more distinct but still overlap: a covector corresponds to a three-dimensional volume, but a bi-tangential vector is represented by a plane same as a bi-covector, since cutting out a 2D plane from four-dimensional space yields a 2D plane again. Only in higher dimensions these symbolic representations become unique. However, both a co-vector and a pseudo-vector will always correspond to (i.e., appear as) an $(n-1)$-dimensional hyperspace.

$$V_\mu dx^\mu \iff v_{\alpha_0\alpha_1\ldots\alpha_{n-1}}\partial_{\alpha_0} \wedge \partial_{\alpha_1} \wedge \ldots \partial_{\alpha_{n-1}} \tag{15}$$

$$v^\mu \partial_\mu \iff V_{\alpha_0\alpha_1\ldots\alpha_{n-1}}dx^{\alpha_0} \wedge dx^{\alpha_1} \wedge \ldots dx^{\alpha_{n-1}} \tag{16}$$

A tangential vector – lhs of (16) – can be understood as one specific direction. Equivalently, it can be seen as "cutting off" all but one $(n-1)$-dimensional hyperspaces from the full n-dimensional space. This equivalence is expressed via the interior product of a tangential vector v with a pseudo-co-scalar Ω yielding a pseudo-covector V (17). Similarly, the interior product of a pseudo-vector with a pseudo-co-scalar yields a tangential vector (17):

$$\iota_\Omega : T(M) \to (T^*)^{(n-1)}(M) : V \mapsto \iota_\Omega v \tag{17}$$

$$\iota_\Omega : T^{(n-1)}(M) \to T^*(M) : V \mapsto \iota_\Omega v \tag{18}$$

Pseudoscalars and pseudo-co-scalars will always be scalar multiples of the basis vectors $\partial_{\alpha 0} \wedge \partial_{\alpha 1} \wedge \ldots \partial_{\alpha n}$ and $dx_0^\alpha \wedge dx_1^\alpha \wedge \ldots dx_n^\alpha$. However, when inversing a coordinate $x^\mu \to -x^\mu$, they flip sign, whereas a "true" scalar does not. An example known from Euclidean vector algebra is the allegedly scalar value constructed from the dot and cross product of three vectors $V(u, v, w) = u \cdot (v \times w)$ which is the negative of when its arguments are flipped:

$$V(u, v, w) = -V(-u, -v, -w) = -u \cdot (-v \times -w) \tag{19}$$

which is actually more obvious when (19) is written as exterior product:

$$V(u, v, w) = u \wedge v \wedge w = V \partial_0 \wedge \partial_1 \wedge \partial_2 \tag{20}$$

The result (20) actually describes the multiple of a volume element span by the basis tangential vectors ∂_μ – any volume must be a scalar multiple of this basis volume element but can flip sign if another convention on the basis vectors is used. This convention depends on the choice of a right-handed versus left-handed coordinate system and is expressed by the orientation tensor $\Omega = \pm \partial_0 \wedge \partial_1 \wedge \partial_2$. In computer graphics, both left-handed and right-handed coordinate systems occur, which may lead to lots of confusions.

By combining (18) and (11) – requiring a metric – we get a map from pseudo-vectors to vectors and reverse. This map is known as the *Hodge star operator* "*":

$$_* : T^{(n-1)}(M) \to T(M) : V \vdash \to \sharp \iota_\Omega V \tag{21}$$

The same operation can be applied to the covectors accordingly and generalized to all vector elements of the exterior algebra on a vector space, establishing a correspondence between $k - vectors$ and $n - k$-vectors. The Hodge star operator allows to identify vectors and pseudo-vectors, similar to how a metric allows to identify vectors and covectors. The Hodge star operator requires a metric and an orientation Ω.

A prominent application in physics using the hodge star operator are the Maxwell equations, which, when written based on the four-dimensional potential $A = V_0 dx^0 + A_k dx^k$ (V_0 the electrostatic, A_k the magnetic vector potential), take the form

$$d_* dA = J \tag{22}$$

with J the electric current and magnetic flow, which is zero in vacuum. The combination $d * d$ is equivalent to the Laplace operator "$\Box$," which indicates that (22) describes electromagnetic waves in vacuum.

Geometric Algebra

Geometric Algebra is motivated by the intention to find a closed algebra on a vector space with respect to multiplication, which includes existence of an inverse operation. There is no concept of dividing vectors in "standard" vector algebra. Neither the inner or outer product has provided vectors of the same dimensionality as their arguments, so they do not provide a closed algebra on the vector space.

Geometric Algebra postulates a product on elements of a vector space $u, v, w \in \mathcal{V}$ that is associative $(uv)w = u(vw)$, left distributive $u(v + w) = uv + uw$, and right distributive $(u + v)w = uw + vw$ and reduces to the inner product as defined by the metric $v^2 = g(v, v)$. It can be shown that the sum of the outer product and the inner product fulfill these requirements; this defines the *geometric product* as the sum of both:

$$uv := u \wedge v + u \cdot v \tag{23}$$

Since $u \wedge v$ and $u \cdot v$ are of different dimensionalities ($\{n\}\ \{\{2\}$ and $\{n\}\ \{\{0\}$, respectively), the result must be in a higher-dimensional vector space of dimensionality $\{n\}\ \{\{2\} + \{n\}\ \{\{0\}$. This space is formed by the linear combination of k-vectors; its elements are called *multivectors*. Its dimensionality is $\sum_{k=0}^{n-1} \begin{pmatrix} n \\ k \end{pmatrix} \equiv 2^n$.

For instance, in two dimensions, the dimension of the space of multivectors is $2^2 = 4$. A multivector V, constructed from tangential vectors on a two-dimensional manifold, is written as

$$V = V^0 + V^1 \partial_0 + V^2 \partial_1 + V^3 \partial_0 \wedge \partial_1 \tag{24}$$

with V^μ the four components of the multivector in a chart. For a three-dimensional manifold, a multivector on its tangential space has $2^3 = 8$ components and is written as

$$\begin{aligned} V = {} & V^0 + \\ & V^1 \partial_0 + V^2 \partial_1 + V^2 \partial_2 + \\ & V^4 \partial_0 \wedge \partial_1 + V^5 \partial_1 \wedge \partial_2 + V^6 \partial_2 \wedge \partial_0 + \\ & V^7 \partial_0 \wedge \partial_1 \wedge \partial_2 \end{aligned} \tag{25}$$

with V^μ the eight components of the multivector in a chart. The components of a multivector have a direct visual interpretation, which is one of the key features of Geometric Algebra. In 3D, a multivector is the sum of a scalar value, three directions, three planes, and one volume. These basis elements span the entire space of multivectors. Geometric Algebra provides intrinsic graphical insight to the algebraic operations. Its application for computer graphics will be discussed in Sect. 4.

Vector and Fiber Bundles

The concept of a fiber bundle data model is inspired by its mathematical correspondence. In short, a fiber bundle is a topological space that looks locally like a product space $B \times F$ of a base space B and a fiber space F.

The *fibers of a function* $f\colon X \to Y$ are the pre-images or inverse images of the points $y \in Y$, i.e., the sets of all elements $x \in X$ with $f(x) = y$:

$$f^{-1}(y) = \{x \in X | f(x) = y\}$$

is a fiber of f (at the point y). A fiber can also be the empty set. The union set of all fibers of a function is called the *total space*. The definition of a fiber bundle makes use of a *projection mappr*$_1$, which is a function that maps each element of a product space to the element of the first space:

$$\begin{aligned} pr_1 : X \times Y &\quad\to\quad X \\ (x, y) &\quad\longmapsto\quad x \end{aligned}$$

Let E, B be topological spaces and $f\colon E \to B$ a continuous map. (E, B, f) is called a *(fiber) bundle* if there exists a space F such that the union of fibers of a neighborhood $U_b \subset B$ of each point $b \in B$ is homeomorphic to $U_b \times F$ such that the projection pr_1 of $U_b \times F$ is U_b again:

$$(E, B, f : E \to B) \text{ bundle} \Longleftrightarrow \exists F : \forall b \in B : \exists U_b : f^{-1}(U_b) \overset{\text{hom}}{\simeq} U_b \times F$$
$$\text{and} \quad pr_1(U_b \times F) = U_b$$

E is called the *total space E*, B is called the *base space*, and $f : E \to B$ the *projection map*. The space F is called the *fiber type* of the bundle or simply the *fiber* of the bundle. In other words, the total space can be written locally as a product space of the base space with some space F. The notation $\mathcal{F}(B) = (E, B, f)$ will be used to denote a fiber bundle over the base space B. It is also said that the *space F fibers over the base space B*.

An important case is the *tangent bundle*, which is the union of all tangent spaces $T_p\ (M)$ on a manifold M together with the manifold $\mathcal{T}(M) := \{(p, v) : p \in M, v \in T_p(M)\}$. Every differentiable manifold possesses a tangent bundle $\mathcal{T}(M)$. The dimension of $\mathcal{T}(M)$ is twice the dimension of the underlying manifold M, its elements are points plus tangential vectors. $T_p\ (M)$ is the fiber of the tangent bundle over the point p.

If a fiber bundle over a space B with fiber F can be written as $B \times F$ globally, then it is called a *trivial bundle* $(B \times F, B, pr_1)$. In scientific visualization, usually only trivial bundles occur. A well-known example for a nontrivial fiber bundle is the Möbius strip.

Topology: Discretized Manifolds

For computational purposes, a topological space is modeled by a finite set of points. Such a set of points intrinsically carries a discrete topology by itself, but one usually considers embeddings in a space that is homeomorphic to Euclidean space to define various structures describing their spatial relationships.

A subset $c \subset X$ of a Hausdorff space X is a *k-cell* if it is homeomorphic to an open k-dimensional ball in $\mathbb{R}^n$. The dimension of the cell is k. Zero-cells are called vertices, one-cells are edges, two-cells are faces or polygons, and three-cells are polyhedra – see also section "Chains." An n-cell within an n-dimensional space is just called a "cell." $(n-1)$-cells are sometimes called "facets" and $(n-2)$-cells are known as "ridges." For k-cells of arbitrary dimension, incidence and adjacency relationships are defined as follows: two cells c_1, c_2 are *incident* if $c_1 \subseteq \partial c_2$, where ∂c_2 denotes the border of the cell c_2. Two cells of the same dimension can never be incident because $\dim(c_1) \neq \dim(c_2)$ for two incident cells c_1, c_2. c_1 is a *side* of c_2 if $\dim(c_1) < \dim(c_2)$, which may be written as $c_1 < c_2$. The special case $\dim(c_1) = \dim(c_2) - 1$ may be denoted by $c_1 \prec c_2$. Two k -cells c_1, c_2 with $k > 0$ are called *adjacent* if they have a common side, i.e.,

$$\text{cell } c_1, c_2 \quad \textbf{adjacent} \Longleftrightarrow \exists \quad \text{cell } f : \ f < c_1, f < c_2$$

For $k = 0$, two zero-cells (i.e., vertices) v_1, v_2 are said to be adjacent if there exists a one-cell (edge) e which contains both, i.e., $v_1 < e$ and $v_2 < e$. Incidence relationships form an incidence graph. A path within an incidence graph is a cell tuple: a *cell-tuple* $\mathcal{C}$ within an n-dimensional Hausdorff space is an ordered sequence of k-cells $(c_n, c_{n-1}, \ldots, c_1, c_0)$ of decreasing dimensions such that $\forall 0 < i \leq n : c_{i-1} \prec c_i$. These relationships allow to determine topological neighborhoods: adjacent cells are called *neighbors*. The set of all $k+1$ cells which are incident to a k-cell forms a neighborhood of the k-cell. The cells of a Hausdorff space X constitute a topological base, leading to the following definition: a ("closure-finite, weak-topology") *CW-complex* $\mathcal{C}$, also called a *decomposition* of a Hausdorff space X, is a hierarchical system of spaces $X^{(-1)} \subseteq X^{(0)} \subseteq X^{(1)} \subseteq \ldots \subseteq X^{(n)}$, constructed by pairwise disjoint open cells $c \subset X$ with the Hausdorff topology $\cup^{c \in \mathcal{C}^c}$, such that $X^{(n)}$ is obtained from $X^{(n-1)}$ by attaching adjacent n-cells to each $(n-1)$-cell and $X^{(-1)} = \emptyset$. The respective subspaces $X^{(n)}$ are called the n-skeletons of X. A CW complex can be understood as a set of cells which are glued together at their subcells. It generalizes the concept of a graph by adding cells of dimension greater than 1.

Up to now, the definition of a cell was just based on a homeomorphism of the underlying space X and $\mathbb{R}^n$. Note that a cell does not need to be "straight," such that, e.g., a two-cell may be constructed from a single vertex and an edge connecting the vertex to itself, as, e.g., illustrated by J. Hart [34]. Alternative approaches toward the definition of cells are more restrictively based on isometry to Euclidean space, defining the notion of "convexity" first. However, it is recommendable to avoid the assumption of Euclidean space and treating the topological properties of a mesh purely based on its combinatorial relationships.

Ontological Scheme and Seven-Level Hierarchy

The concept of the fiber bundle data model builds on the paradigm that numerical data sets occurring for scientific visualization can be formulated as trivial fiber bundles (see section "Vector and Fiber Bundles"). Hence, data sets may be distinguished by their properties in the base space and the fiber space. At each point of the – discretized – base space, there are some data in the fiber space attached. Basically a fiber bundle is a set of points with neighborhood information attached to each of them. An n-dimensional array is a very simple case of a fiber bundle with neighborhood information given implicitly.

The structure of the base space is described as a CW complex, which categorizes the topological structure of an n-dimensional base space by a sequence of k-dimensional skeletons, with $0 < k < n$. These skeletons carry certain properties of the data set: the zero-skeleton describes vertices, the one-skeleton refers to edges, two-skeleton to the faces, etc., of some mesh (a triangulation of the base space). Structured grids are triangulations with implicitly given topological properties. For instance, a regular n-dimensional grid is one where each point has 2^n neighbors.

The structure of the fiber space is (usually) not discrete and given by the properties of the geometrical object residing there, such as a scalar, vector, covector, and tensor. Same as the base space, the fiber space has a specific dimensionality, though the dimensionality of the base space and fiber space is independent. Figure 3 demonstrates example images from scientific visualization classified via their fiber bundle structure. If the fiber space has vector space properties, then the fiber bundle is a vector bundle and vector operations can be performed on the fiber space, such as addition, multiplication, and derivation.

The distinction between base space and fiber space is not common use in computer graphics, where topological properties (base space) are frequently intermixed with geometrical properties (coordinate representations). Operations in the fiber space can, however, be formulated independently from the base space, which leads to a more reusable design of software components. Coordinate information, formally part of the base space, can as well be considered as fiber, leading to further generalization. The data sets describing a fiber are ideally stored as contiguous arrays in memory or disk, which allows for optimized array and vector operations. Such a storage layout turns out to be particularly useful for communicating data with the GPU using vertex buffer objects: the base space is given by vertex arrays (e.g., OpenGL *glVertexPointer*), and fibers are attribute arrays (e.g., OpenGL *glVertexAttribPointer*), in the notation of computer graphics. While the process of hardware rendering in its early times had been based on procedural descriptions (cached in display lists), vertex buffer objects are much faster in state-of-the-art technology. Efficient rendering routines are thus implemented as *maps* from fiber bundles in RAM to fiber bundles in GPU memory (eventually equipped with a GPU shader program).

A complex data structure (such as some color-coded time-dependent geometry) will be built from many data arrays. The main question that needs to be answered by

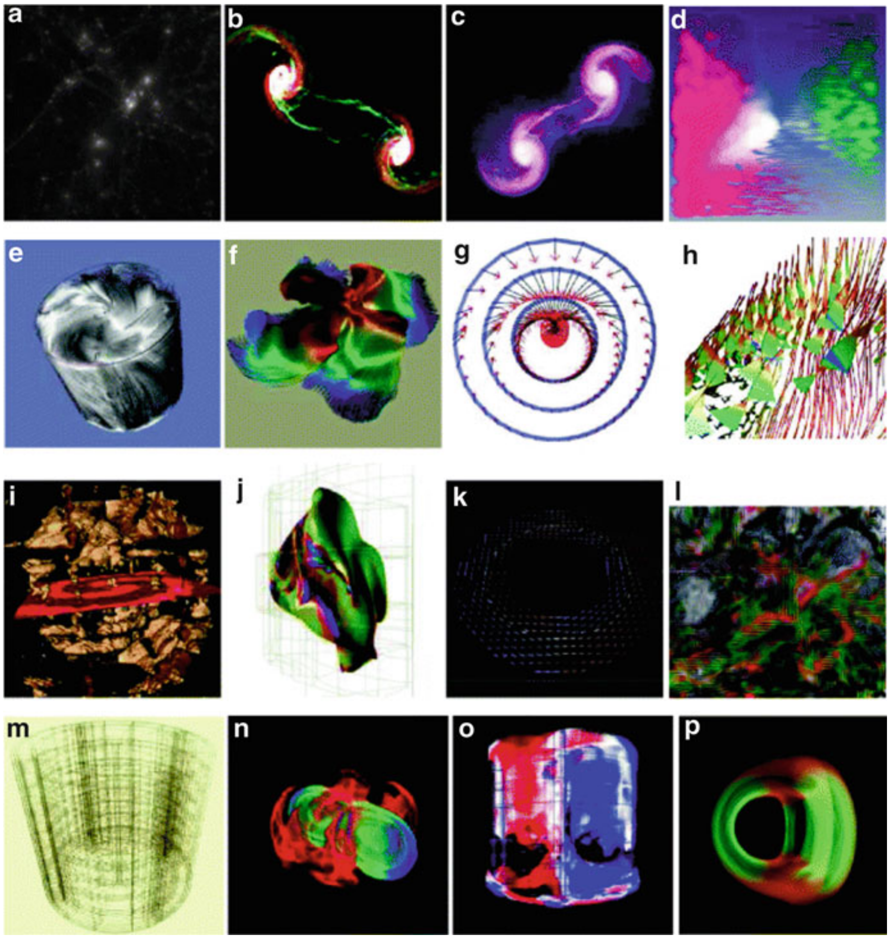


Fig. 3 Fiber bundle classification scheme for visualization methods:dimensionality of the base space (involving the k-skeleton of the discretized manifold) and dimensionality of the fiber space (involving the number of field quantities per element, zero referring to display of the mere topological structure). (**a**) Zero-cells, 0D. (**b**) Zero-cells, 1D. (**c**) Zero-cells, 3D. (**d**) Zero-cells, 6D. (**e**) One-cells, 0D. (**f**) One-cells, 1D. (**g**) One-cells, 3D. (**h**) One-cells, 6D. (**i**) Two-cells, 0D. (**j**) Two-cells, 1D. (**k**) Two-cells, 3D. (**l**) Two-cells, 6D. (**m**) Three-cells, 0D. (**n**) Three-cells, 1D. (**o**) Three-cells, 3D. (**p**) Three-cells, 6D

a data model is how to assign a semantic meaning to each of these data arrays – what do the numerical values actually *mean*? It is always possible to introduce a set of keywords with semantics attached to them. In addition, the introduction of keywords also reduces the number of possible identifiers available for user-specific purpose. This problem is also known as "name space pollution". The approach followed in the data model presented in [7] is to avoid use of keywords as much as possible. Instead, it assigns the semantics of an element of the data structure into the placement of this element. The objective is to describe all data types that occur in an algorithm (including file reader and rendering routines) within this model. It is formulated as a graph of up to seven levels (two of them optional). Each level represents a certain property of the entire data set, the *Bundle*. These levels are called:

1. *Slice*
2. *Grid*
3. *Skeleton*
4. *Representation*
5. *Field*
6. (Fragment)
7. (Compound Elements)

Actual data arrays are stored only below the "*Field*" level. Given one hierarchy level, the next one is accessed via some identifier. The type of this identifier differs for each level: numerical values within a *Skeleton* level are grouped into *Representation* objects, which hold all information that is *relative* to a certain "representer." Such a representer may be a coordinate object that, for instance, refers to some Cartesian or polar chart, or it may well be another *Skeleton* object, either within the same *Grid* object or even within another one. An actual data set is described through the existence of entries in each level. Only two of these hierarchy levels are exposed to the end user; these are the *Grid* and *Field* levels. Their corresponding textual identifiers are arbitrary names specified by the user.

Hierarchy object	Identifier type	Identifier semantic
Bundle	Floating point number	Time value
Slice	String	Grid name
Grid	Integer set	Topological properties
Skeleton	Reference	Relationship map
Representation	String	Field name
Field	Multidimensional index	Array index

A *Grid* is subset of data within the Bundle that refers to a specific geometrical entity. A *Grid* might be a mesh carrying data such as a triangular surface, a data cube, a set of data blocks from a parallel computation, or many other data types. A *Field* is the collection of data sets given as numbers on a specific topological component of a Grid, for instance, floating point values describing pressure or temperature on a Grid's vertices. All other levels of the data model describe the properties of the Bundle as construction blocks. The usage of these construction blocks constitutes a certain language to describe data sets. A Slice is identified by a single floating point number representing time (generalization to arbitrary-dimensional parameter spaces is possible). A *Skeleton* is identified by its dimensionality, index depth (relationship to the vertices of a Grid), and refinement

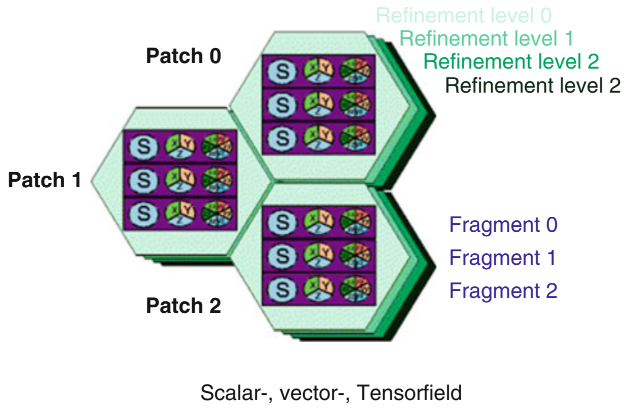


Fig. 4 Hierarchical structure of the data layout of the concept of a *field* in computer memory: (1) organization by multiple resolutions for same spatial domain; (2) multiple coordinate systems covering different spatial domains (arbitrary overlap possible); (3) fragmentation of fields into blocks (recombination from parallel data sources); and (4) layout of compound fields as components for performance reasons, indicated as S (scalar field), $\{x, y, z\}$ for vector fields, and $\{xx, xy, yy, yz, zz, zx\}$ for tensor fields

level. This will be explained in more detail in section "Topological Skeletons." The scheme also extends to cases beyond the purely mathematical basis to also cover data sets that occur in praxis, which is described in section "Non-topological representations." A representation is identified via some reference object, which may be some coordinate system or another *Skeleton*. The lowest levels of fragments and compounds describe the internal memory layout of a *Field* data set and are optional; some examples are described in [8,9].

Field Properties

A specific *Field* identifier may occur in multiple locations. All these locations together define the properties of a field. The following four properties are expressible in the data model:

1. *Hierarchical ordering*: For a certain point in space, there exist multiple data values, one for each *refinement* level. This property describes the topological structure of the base space.
2. *Multiple coordinate systems*: One spatial point may have multiple data representations relating to different coordinate systems. This property describes the geometrical structure of the base space.
3. *Fragmentation*: Data may stem from multiple sources, such as a distributed multiprocess simulation. The field then consists of multiple data blocks, each of them covering a subdomain of the field's base space. Such field fragments may also overlap, known as "ghost zones."
4. *Separated Compounds*: A compound data type, such as a vector or tensor, may be stored in different data layouts since applications have their own preferences. An

array of tensors may also be stored as a tensor of arrays, e.g., *XYZXYZXYZXYZ* as *XXXXYYYYZZZZ*. This property describes the internal structure of the fiber space.

All of these properties are optional. In the most simple case, a field is just represented by an array of native data types; however, in the most general case (which the visualization algorithm must always support), the data are distributed over several such property elements and built from many arrays. With respect to quick transfer to the GPU, only the ability to handle multiple arrays per data set is of relevance.

Figure 4 illustrates the organization of the last four levels of the data model. These consist of Skeleton and Representation objects with optional fragmentation and compound levels. The ordering of these levels is done merely based on their semantic importance, with the uppermost level (1) embracing multiple resolutions of the spatial domain being the most visible one to the end user. Each of these resolution levels may come with different topological properties, but all arrays within the same resolution are required to be topologically compatible (i.e., share the same number of points). There might still be multiple coordinate representations required for each resolution, which constitutes the second hierarchy level (2) of multiple coordinate patches. Data per patch may well be distributed over various fragments (3), which is considered an internal structure of each patch, due to parallelization or numerical issues, but not fundamental to the physical setup. Last but not least, fields of multiple components such as vector or tensor fields may be separated into distinct arrays themselves [7]. This property, merely a performance issue of in-memory data representation, is not what the end user usually does not want to be bothered with and is thus set as the lowest level in among these four entries.

Topological Skeletons

The *Skeleton* level of the fiber bundle hierarchy describes a certain topological property. This can be the vertices, the cells, the edges, etc. Its primary purpose is to describe the skeletons of a CW complex, but they may also be used to specify mesh refinement levels and agglomerations of certain elements. All data fields that are stored within a *Skeleton* level provide the same number of elements. In other words they share their index space (a data space in HDF5 terminology). Each *Topology* object within a *Grid* object is uniquely identified via a set of integers, which are the *dimension* (e.g., the dimension of a k-cell), *index depth* (how many dereferences are required to access coordinate information in the underlying manifold), and *refinement level* (a multidimensional index, in general). Vertices – index depth 0 – of a topological space of dimension n define a Skeleton of type $(n, 0)$. Edges are one-dimensional sets of vertex indices; therefore, their index depth is 1 and their Skeleton type is (1,1). Faces are two-dimensional sets of vertex indices, hence Skeleton type (2, 1). Cells – such as a tetrahedron or hexahedra – are described by a Skeleton type (3, 1).

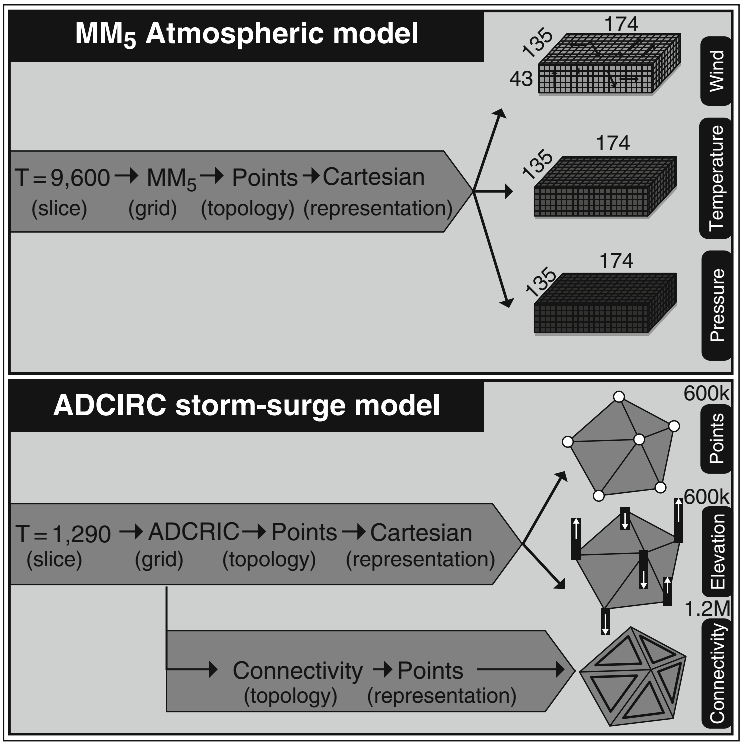

MM5 Atmospheric model
T = 9,600 → MM5 → Points → Cartesian
(slice) (grid) (topology) (representation)
174
135
43
Wind
Temperature
Pressure
ADCIRC storm-surge model
T = 1,290 → ADCRIC → Points → Cartesian
(slice) (grid) (topology) (representation)
Connectivity → Points
(topology) (representation)
600k
600k
1.2M
Points
Elevation
Connectivity

Fig. 5 The five-level organization scheme used for atmospheric data (MM5 model data) and surge data (ADCIRC simulation model), built upon common topological property descriptions with additional fields (From Venkataraman et al. [81])

All the Skeleton objects of index depth 1 build the k-skeletons of a manifold's triangulation.

Higher index depths describe sets of k-cells. For instance, a set of edges describes a line – a path along vertices in a Grid. Such a collection of edges will fit into a Skeleton of dimension 1 and depth 2, i.e., type (1, 2). It is a one-dimensional object of indices that refer to edges that refer to vertices.

Non-topological Representations

Polynomial coordinates, information on field fragments, histograms, and color maps can be formulated in the fiber bundle model as well. These quantities are no longer direct correspondences of the mathematical background, but they may still be cast into the given context.

Coordinates may be given procedurally, such as via some polynomial expression. The data for such expressions may be stored in a Skeleton of *negative* index depth – as these data are required to compute the vertex coordinates and more fundamental than these in this case.

A *fragment* of a Field given on vertices – the $(n, 0)$-Skeleton of a Grid – defines an n-dimensional subset of the Grid, defined by the hull of the vertices corresponding to the fragments. These may be expressed as a $(n, 2)$-Skeleton, where the Field object named "Positions" (represented relative to the vertices) refers to the (global) vertex indices of the respective fragments. The representation in coordinates corresponds to its range, known as the *bounding box*. Similarly, a field given on the vertices will correspond to the field's numerical *minimum/maximum range* within this fragment.

A *histogram* is the representation of a field's vertex complex in a "chart" describing the required discretization, depending on the min/max range and a number count. A *color map* (transfer function) can be interpreted as a chart object itself. It has no intrinsically geometrical meaning, but provides means to transform some data. For instance, some scalar value will be transformed to some RGB triple using some color map. A scalar field represented in a certain color map is therefore of type RGB values and could be stored as an array of RGB values for each vertex. In practice, this will not be done since such transformation is performed in real time by modern graphics hardware. However, this interpretation of a color map as a chart object tells how color maps may be stored in the fiber bundle data model.

3 Differential Forms and Topology

This section not only introduces the concepts of differential forms and their discrete counterparts but also illustrates that similar concepts are applied in several separate areas of scientific visualization. Since the available resources are discrete and finite, concepts mirroring these characteristics have to be applied to visualize complex data sets. The most distinguished algebraic structure is described by exterior algebra (or Grassmann algebra, see also section "Exterior Product"), which comes with two operations, the exterior product (or wedge product) and the exterior derivative.

Differential Forms

Manifolds can be seen as a precursor to model physical quantities of space. Charts on a manifold provide coordinates, which allows using concepts which are already well established. Furthermore, they are crucial for the field of visualization, as they are key components to obtain depictable expressions of abstract entities. Tangential vectors were already introduced in section "Tangential Vectors" as derivatives along a curve. Then a one-form α is defined as a linear mapping which assigns a value to each tangential vector v from the tangent space $T_P(M)$, i.e., $\alpha : T_P(M) \to \mathbb{R}$. They are commonly called co-variant vectors, covectors (see section "Tangential Vectors"), or Pfaff-forms. The set of one-forms generates the dual vector space or cotangential space $T_p^*(M)$. It is important to highlight that the tangent vectors $v \in T_P(M)$ are not contained in the manifold itself, so the differential forms also generate an additional space over $P \in M$. In the following, these one-forms are generalized to (alternating) differential forms.

An alternative point of view treats a tangential vector v as a linear mapping which assigns a scalar to each one-form α by $< \alpha, v > \in \mathbb{R}$. By omitting one of the arguments of the obtained mappings, $< \alpha, . >$ or $\alpha(v)$ and $<., v >$ or $v(\alpha)$, linear objects are defined. Multi-linear mappings depending on multiple vectors or covectors appear as an extension of this concept and are commonly called tensors

$$\gamma : T^{*m} \times T^{n} \to \mathbb{R} \tag{26}$$

where n and m are natural numbers and T^n and T^{*m} represent the n and m powered Cartesian product of the tangential space or the dual vector space (cotangential space). A tensor γ is called an (n, m)-tensor which assigns a scalar value to a set of m covectors and n vectors. All tensors of a fixed type (n, m) generate a tensor space attached at the point $P \in M$. The union of all tensor spaces at the points $P \in M$ is called a *tensor bundle*. The tangential and cotangential bundles are specialized cases for (1, 0) and (0, 1) tensor bundles, respectively. Fully antisymmetric tensors of type $(0, m)$ may be identified with *differential forms of degree m*. For $m >$ dim(M), where dim(M) represents the dimension of the manifold, differential forms vanish.

The *exterior derivative* or Cartan derivative of differential forms generates a $p+$ 1-form df from a p-form f and conforms to the following requirements:

1. Compatibility with the wedge product (product rule): $d(\alpha \wedge \beta) = d\alpha \wedge \beta + (-1)^m \alpha \wedge d\beta$
2. Nilpotency of the operation d, $d \circ d = 0$, depicted in Fig. 11
3. Linearity

   A subset of one-forms is obtained as a differential df of zero-forms (functions) f at P and are called *exact differential forms*. For an n-dimensional manifold M, a one-form can be depicted by drawing $(n-1)$-dimensional surfaces, e.g., for the three-dimensional space, Fig. 6 depicts a possible graphical representation of a one-form attached to M. This depiction also enables a graphical representation on how to integrate differential forms, where only the number of surfaces which are intersected by the integration domain has to be counted:

$$< df, v >= df(v) = \alpha(v) \tag{27}$$

A consequence of being exact includes the closeness property $d\alpha = 0$. Furthermore, the integral $\int_{C_p} df$ with C_p representing an integration domain, e.g., an interval x_1 and x_2, results in the same value $f(x_2) - f(x_1)$. In the general case, a p-form is not always the exterior derivative of a p-one-form; therefore, the integration of p-forms is not independent of the integration domain. An example is given by the exterior derivative of a p-form β resulting in a $p + 1$-form $\gamma = d\beta$. The structure of such a generated differential form can be depicted by a tube-like structure such as in Fig. 7. While the wedge product of an r-form and an s-form results in an $r + s$-form, this resulting form is not necessarily representable as a derivative. Figure 7 depicts a two-form which is not constructed by the exterior

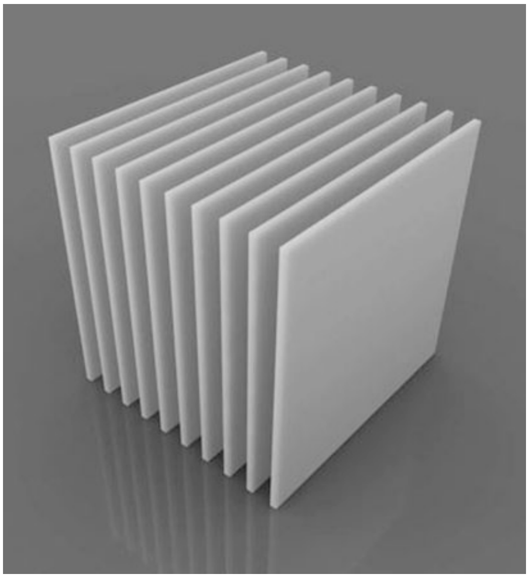


Fig. 6 Possible graphical representation of the topological structure of one-forms in three dimensions. Note that the graphical display of differential forms varies in different dimension and does not depend on the selected basis elements

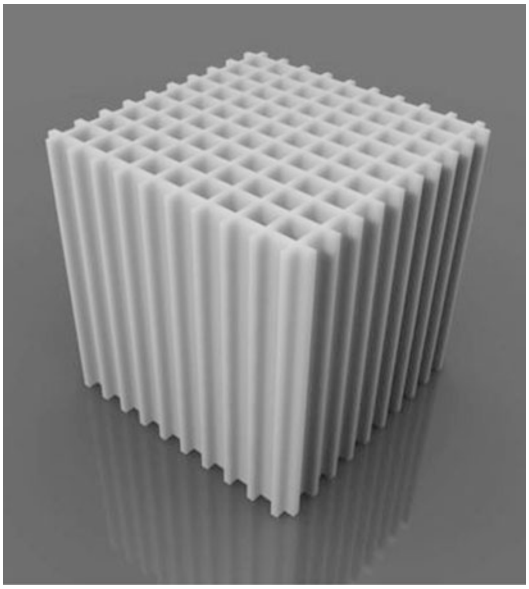


Fig. 7 Possible graphical representation of a general two-form generated by $\alpha \wedge \beta$, where α and β are one-forms. The topologically tube-like structure of the two-forms is enclosed by the depicted planes

derivative but instead by $\alpha \wedge \beta$, where α and β are one-forms. In the general case, a p-form attached on an n-dimensional manifold M is represented by using $(n - p)$-dimensional surfaces.

By sequentially applying the operation d to $(0, m)$ for $0 \leq m \leq \dim(M)$, the *de Rham complex* is obtained, which enables the investigation of the relation of closed and exact forms. The de Rham complex enables the transition from the continuous differential forms to the discrete counterpart, so-called cochains. The already briefly mentioned topic of integration of differential forms is now mapped onto the integration of these cochains. To complete the description, the notion of chains, also modeled by multivectors (as used in Geometric Algebra, see section "Geometric Algebra" and Sect. 4) or fully antisymmetric $(n, 0)$-tensors, as description of integration domains is presented, where a chain is a collection of n-cells.

The connection between chains and cochains is investigated in algebraic topology under the name of homology theory, where chains and cochains are collected in additive Abelian groups $C_p\ (M)$.

Chains

The de Rham complex collects cochains similar to a cell complex aggregating cells as elements of chains. To use these elements, e.g., all edges, in a computational manner, a mapping of the n-cells onto an algebraic structure is needed. An algebraic representation of the assembly of cells, an n-chain, over a cell complex $\mathcal{K}$ and a vector space $\mathcal{V}$ can be written by

$$c_n = \sum\nolimits_{i=1}^{j} w_i \tau_n^i \quad \tau_n^i \in \mathcal{K},\ w_i \in \mathcal{V}$$

which is closed under reversal of the orientation:

$$\forall \tau_n^i \in c_n \quad \text{there is} - \tau_n^i \in c_n$$

The different topological elements are called cells, and the dimensionality is expressed by adding the dimension such as a three-cell for a volume, a two-cell for surface elements, a one-cell for lines, and a zero-cell for vertices. If the coefficients are restricted to $\{-1, 0, 1\} \in \mathbb{Z}$, the following classification for elements of a cell complex is obtained:

- 0: if the cell is not in the complex
- 1: if the unchanged cell is in the complex
- -1 : if the orientation is changed

The so-called boundary operator is a map between sets of chains C_p on a cell complex K. Let us denote the ith p-cell as $\tau_p^i = k_0, \ldots k_p$, whereby $\tau_p^i \in K$. The boundary operator ∂_p defines a $(p-1)$-chain computed from a p-chain: $\partial_p : C_p(K) \dashrightarrow C_{p-1}(K)$. The boundary of a cell τ_p^j can be written as alternating sum over elements of dimension $p - 1$:

$$\partial_p \tau_p^i = \sum\nolimits_i (-1)^i [k_0, k_1, \ldots, \tilde{k}_i, \ldots k_n] \tag{28}$$

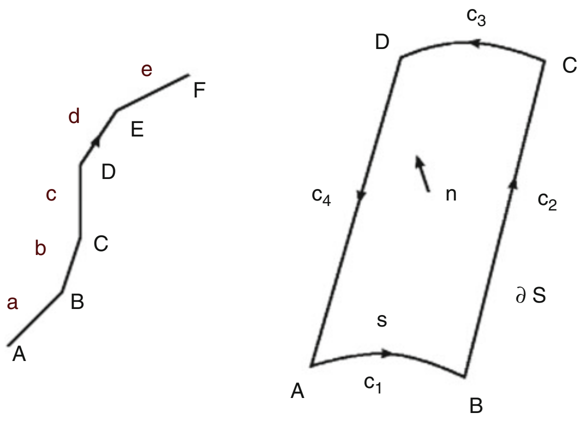


Fig. 8 Representation of a one-chain τ_1^i with zero-chain boundary τ_0^j (*left*) and a two-chain τ_2 with one-chain boundary τ_1^k (*right*)

where $\tilde{k_i}$ indicates that k_i is deleted from the sequence. This map is compatible with the additive and the external multiplicative structure of chains and builds a linear transformation:

$$C_p \to C_{p-1} \tag{29}$$

Therefore, the boundary operator is linear

$$\partial \left(\sum_i w_i \tau_p^i \right) = \sum_i w_i \left(\partial \tau_p^i \right) \tag{30}$$

which means that the boundary operator can be applied separately to each cell of a chain. Using the boundary operator on a sequence of chains of different dimensions results in a chain complex $C_* = \{C_p, \partial_p\}$ such that the complex property

$$\partial_{p-1} \partial_p = 0 \tag{31}$$

is given. Homological concepts are visible here for the first time, as homology examines the connectivity between two immediately neighboring dimensions. Figure 8 depicts two examples of one-chains and two-chains and an example of the boundary operator.

Applying the appropriate boundary operator to the two-chain example reads

$$\begin{gathered} \partial_2 \tau_2 = \tau_1^1 + \tau_1^2 + \tau_1^3 + \tau_1^4 \\ \partial_1(\tau_1^1 + \tau_1^2 + \tau_1^3 + \tau_1^4) = \tau_0^1 + \tau_0^2 - \tau_0^2 + \tau_0^3 - \tau_0^3 + \tau_0^4 - \tau_0^4 - \tau_0^1 = 0 \end{gathered} \tag{33}$$

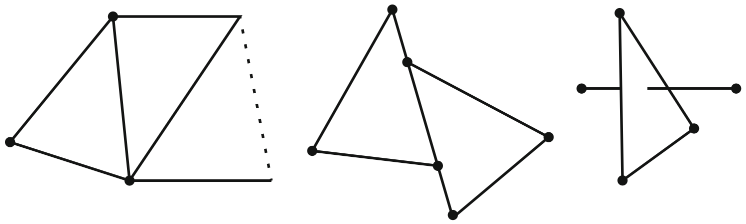


Fig. 9 Examples of violations of correct cell attachment. *Left*: missing zero-cell. *Middle*: cells do not intersect at vertices. *Right*: intersection of cells

A different view on chain complexes presents itself when the main focus is placed on the cells within a chain. To cover even the most abstract cases, a cell is defined as a subset $c \subset X$ of a Hausdorff space X if it is homeomorphic to the interior of the open n-dimensional ball $\mathbb{D}^n = \{x \in \mathbb{R}^n : |x| < 1\}$. The number n is unique due to the *invariance of domain* theorem [13] and is called the dimension of c, whereas homeomorphic means that two or more spaces share the same topological characteristics. The following list assigns terms corresponding to other areas of scientific computing:

- 0-cell: point
- 1-cell: edge
- 2-cell: facet
- n-cell: cell

A cell complex $\mathcal{K}$ (see also section "Topology: Discretized Manifolds") can be described by a set of cells that satisfy the following properties:

- The boundary of each p-cell τ_p^i is a finite union of $(p-1)$-cells in $\mathcal{K} : \partial_p \tau_p^i = \bigcup^m \tau_{p-1}^m$.
- The intersection of any two cells τ_p^i, τ_p^j in $\mathcal{K}$ is either empty or is a unique cell in $\mathcal{K}$.

The result of these operations are subspaces $X^{(n)}$ which are called the n-skeletons of the cell complex. Incidence and adjacence relations are then available. Examples for incidence can be given by vertex on edge relation and for adjacency by vertex to vertex relations. This cell complex with the underlying topological space guarantees that all interdimensional objects are connected in an appropriate manner. Although there are various possible attachments of cells, only one process results in a cell complex, see Fig. 9.

Cochains

In addition to chain and cell complices, scientific visualization requires the notation and access mechanisms to global quantities related to macroscopic n-dimensional space-time domains. The differential forms which are necessary concepts to handle physical properties can also be projected onto discrete counterparts, which are called *cochains*. This collection of possible quantities, which can be measured, can then be called a section of a fiber bundle, which permits the modeling of these measurements as a function that can be integrated on arbitrary n-dimensional (sub)domains or multivectors. This function can then be seen as the abstracted process of measurement of this quantity [55, 75]. The concept of cochains allows the association of numbers not only to single cells, as chains do, but also to assemblies of cells. Briefly, the necessary requirements are that this mapping is not only orientation dependent but also linear with respect to the assembly of cells. A cochain representation is now the global quantity association with subdomains of a cell complex, which can be arbitrarily built to discretize a domain.

A linear transformation σ of the n-chains into the field $\mathbb{R}$ of real numbers forms a vector space $c_n \longrightarrow \wedge\{\sigma\}\mathbb{R}$ and is called a vector-valued m-dimensional cochain or short m-cochain. The coboundary δ of an m-cochain is an $(m+1)$-cochain defined as

$$\delta c^m = \sum\nolimits_i v_i \tau_i, \quad \text{where} \quad v_i = \sum\nolimits_{b \,\in\, \text{faces}(\tau_i)} \sigma(b, \tau_i) c_m(b) \tag{34}$$

Thus, the coboundary operator assigns nonzero coefficients only to those $(m+1)$ cells that have c_m as a face. As can be seen, δc_m depends not only on c_m but on how c_m lies in the complex $\mathcal{K}$. This is a fundamental difference between the two operators ∂ and δ. An example is given in Fig. 10 where the coboundary operator is used on a one-cell. The right part $\delta \circ \delta\mathcal{K}$ of Fig. 10 is also depicted for the continuous differential forms in Fig. 7. The coboundary of an m-cochain is an $(m+1)$-cochain which assigns to each $(m+1)$ cell the sum of the values that the $m+1$-cochain assign to the m-cells which form the boundary of the $(m+1)$ cell. Each quantity appears in the sum multiplied by the corresponding incidence number. Cochain complices [33, 35] are similar to chain complices except that the arrows are reversed, so a cochain complex $C^* = \{C^m, \delta^m\}$ is a sequence of modules C^m and homomorphisms:

$$\delta^m : C^m \to C^{m+1} \tag{35}$$

such that

$$\delta^{m+1}\delta^m = 0 \tag{36}$$

$\mathcal{K}^1 \xrightarrow{\delta} \delta\mathcal{K}^1 \xrightarrow{\delta} \delta \circ \delta\mathcal{K}^1 = 0$. Proceeding from left to right, a one-cochain represented by a line segment, a two-cochain generated by the product of two one-forms, and a three-cochain depicted by volume objects are illustrated.

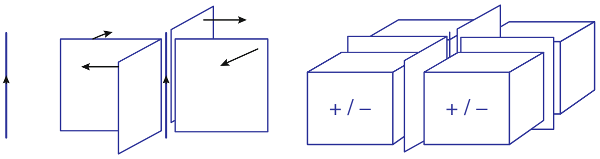


Fig. 10 Cochain complex with the corresponding coboundary operator

Then, the following sequence with $\delta \circ \delta = 0$ is generated:

$$0 \xrightarrow{\delta} C^0 \xrightarrow{\delta} C^1 \xrightarrow{\delta} C^2 \xrightarrow{\delta} C^3 \xrightarrow{\delta} 0 \tag{37}$$

Cochains are the algebraic equivalent of alternating differential forms, while the coboundary process is the algebraic equivalent of the external derivative and can therefore be considered as the discrete counterpart of the differential operators:

- grad.
- curl.
- div.

It indeed satisfies the property $\delta \circ \delta \equiv 0$ corresponding to

- curlgrad. $\equiv 0$
- divcurl. $\equiv 0$

Duality Between Chains and Cochains

Furthermore, a definition of the adjoint nature of $\partial, \delta : C^p \to C^{p+1}$ can be given:

$$\langle c^p, \partial c_{p+1} \rangle = \langle \delta c^p, c_{p+1} \rangle \tag{38}$$

The concepts of chains and cochains coincide on finite complices [45]. Geometrically, however, C_p and C^p are distinct [12] despite an isomorphism. An element of C_p is a formal sum of p-cells, where an element of C^p is a linear function that maps elements of C_p into a field. Chains are dimensionless multiplicities of aggregated cells, whereas those associated with cochains may be interpreted as physical quantities [65]. The extension of cochains from single cell weights to quantities associated with assemblies of cells is not trivial and makes cochains very different from chains, even on finite cell complices. Nevertheless, there is an important duality between p-chains and p-cochains. The first part of the de Rham (cohomology group) complex, depicted in Fig. 11 on the left, is the set of closed one-forms modulo the set of exact one-forms denoted by

$$H^1 = Z^1/B^1 \tag{39}$$

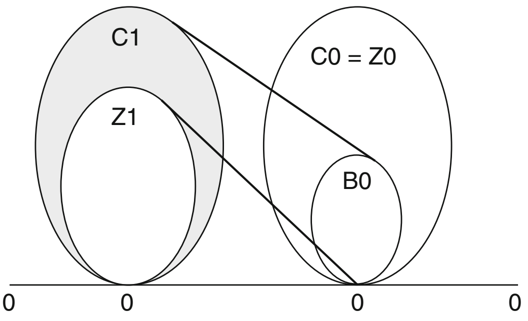


Fig. 11 A graphical representation of closed and exact forms. The forms Z_1, B_1, Z_0, and B_0 are closed forms, while only the forms B_0 and B_1 are exact forms. The nilpotency of the operation d forces the exact forms to vanish

This group is therefore trivial (only the zero element) if all closed one-forms are exact. If the corresponding space is multiply connected, then there are closed one-chains that are not themselves boundaries, and there are closed one-forms that are not themselves exact.

For a chain $c_p \in C_p(\mathcal{K}, \mathbb{R})$ and a cochain $c^p \in C^p(\mathcal{K}, \mathbb{R})$, the integral of c^p over c_p is denoted by $\int_{c_p} c^p$, and integration can be regarded as a mapping, where D represents the corresponding dimension:

$$\int : C_p(\mathcal{K}) \times C^p(\mathcal{K}) \to \mathbb{R}, \qquad \text{for } 0 \leq p \leq D \tag{40}$$

Integration in the context of cochains is a linear operation: given $a_1, a_2 \in \mathbb{R}$, $c^{p,1} c^{p,2} \in C^p(\mathcal{K})$ and $c_p \in C_p(\mathcal{K})$, reads

$$\int_{c_p} a_1 c^{p,1} + a_2 c^{p,2} = a_1 \int_{c_p} c^{p,1} + a_2 \int_{c_p} c^{p,2} \tag{41}$$

Reversing the orientation of a chain means that integrals over that chain acquire the opposite sign

$$\int_{-c_p} c^p = - \int_{c_p} c^p \tag{42}$$

using the set of p-chains with vector space properties $C_p(\mathcal{K}, \mathbb{R})$, e.g., linear combinations of p-chains with coefficients in the field $\mathbb{R}$. For coefficients in $\mathbb{R}$, the operation of integration can be regarded as a bilinear pairing between p-chains and p-cochains. Furthermore, for reasonable p-chains and p-cochains, this bilinear pairing for integration is nondegenerate,

$$\text{if} \quad \int_{c_p} c^p = 0 \quad \forall c_p \in C_p(\mathcal{K}), \quad \text{then } c^p = 0 \tag{43}$$

and

$$\text{if} \quad \int_{c_p} c^p = 0 \quad \forall c^p \in C^p(\mathcal{K}), \quad \text{then } c_p = 0 \tag{44}$$

The integration domain can be described by, using Geometric Algebra notation, the exterior product applied to multivectors. An example is then given by the generalized Stokes theorem:

$$\int_{c_p} df = \int_{\partial c_p} f \tag{45}$$

or

$$< df, c_p > = < f, \partial c_p > \tag{46}$$

The generalized Stokes theorem combines two important concepts, the integration domain and the form to be integrated.

Homology and Cohomology

The concepts of chains can also be used to characterize properties of spaces, the homology and cohomology, where it is only necessary to use $C_p(\mathcal{K}, \mathbb{Z})$. The algebraic structure of chains is an important concept, e.g., to detect a p-dimensional hole that is not the boundary of a $p + 1$-chain, which is called a p-cycle. For short, a cycle is a chain whose boundary is $\partial_p c_p = 0$, a closed chain. The introduced boundary operator can also be related to homological terms. A boundary is a chain b_p for which there is a chain c_p such that $\partial_p c_p = b_p$. Since $\partial \circ \partial = 0$, $B_n \subset Z_n$ is obtained. The homology is then defined by $H_n = Z_n / B_n$. The homology of a space is a sequence of vector spaces. The topological classification of homology is defined by

$$\begin{aligned} B_p &= \text{im } \partial_{p+1} \qquad \text{and} \\ Z_p &= \text{ker } \partial_p \end{aligned}$$

so that $B_p \subset Z_p$ and

$$H_p = Z_p / B_p$$

where $\beta_p = \{\text{Rank}\}\ H_p$. Here $\{\text{im}\}$ is the image and $\{\text{ker}\}$ is the kernel of the mapping. For cohomology

$$\begin{aligned} B^p &= \text{im } d^{p+1} \qquad \text{and} \\ Z^p &= \text{ker } d^p \end{aligned}$$

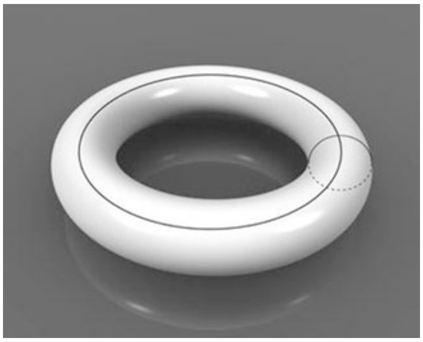


Fig. 12 Topologically a torus is the product of two circles. The partially shaded circle is spun around the fully drawn circle which can be interpreted as the closure of a cylinder onto itself

so that $B^p \subset Z^p$ and

$$H^p = Z^p / B^p$$

where $\beta^p = \{\text{Rank}\}\ H^p$. An important property of these vector spaces is given by β, which corresponds to the dimension of the vector spaces H and is called the Betti number [35, 86]. Betti numbers identify the number of nonhomologous cycles which are not boundaries:

- β_0 counts the number of connected components.
- β_1 counts the number of tunnels (topological holes).
- β_2 counts the number of enclosed cavities.

The number of connected components gives the number of distinct entities of a given object, whereas tunnels describe the number of separated parts of space. In contrast to a tunnel, the enclosed cavities are completely bounded by the object.

Examples for the Betti numbers of various geometrical objects are stated by:

- Cylinder: $\beta_0 = 1, \beta_1 = 1, \beta_n = 0 \forall n \geq 2$. The cylinder consists of one connected component, which forms a single separation of space. Therefore no enclosed cavitiy is present.
- Sphere: $\beta_0 = 1, \beta_1 = 0, \beta_2 = 1, \beta_n = 0 \forall n \geq 3$. If β_1 and β_2 are switched, a sphere is obtained by contracting the separation by generating an enclosed cavity from the tunnel.
- Torus: $\beta_0 = 1, \beta_1 = 2, \beta_2 = 1, \beta_n = 0 \forall n \geq 3$. Closing a cylinder onto itself results in a torus which not only generates an enclosed cavity but also maintains the cylinder's tunnel. An additional tunnel is introduced due to the closing procedure which is depicted in Fig. 12 as the central hole.

The Euler characteristics, which is an invariant, can be derived from the Betti numbers by: $\xi = \beta_0 - \beta_1 + \beta_2$.

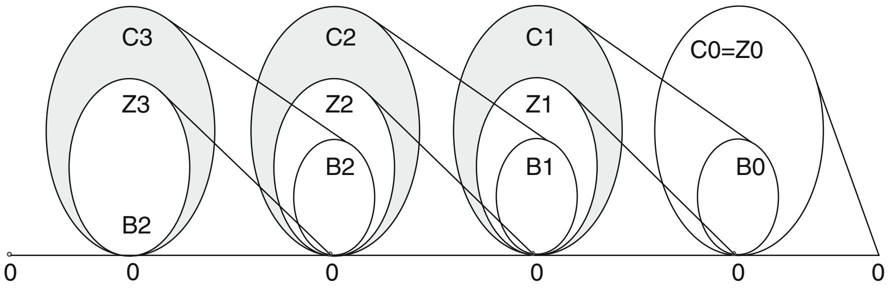


Fig. 13 A graphical representation of (co)homology for a three-dimensional cell complex

Fig. 14 Illustration of cycles A, B, C and a boundary C. A, B are not boundaries

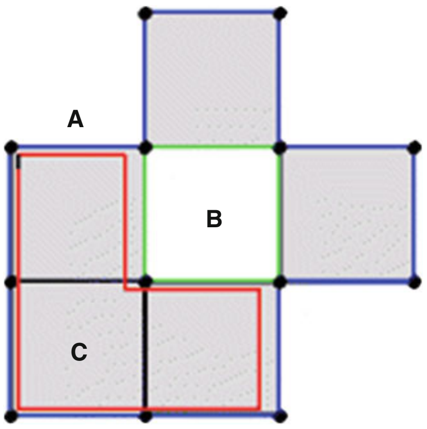


Figure 13 depicts the homology of a three-dimensional chain complex with the respective images and kernels, where the chain complex of $\mathcal{K}$ is defined by $\{\text{im}\}\ \partial_{p+1} \subseteq \{\text{ker}\}\ \partial_p$. As can be seen, the boundary operator expression yields $\partial_p \circ \partial_{p+1} = 0$.

To give an example, the first homology group is the set of closed one-chains (curves) modulo the closed one-chains which are also boundaries. This group is denoted by $H_1 = Z_1/B_1$, where Z_1 are cycles or closed one-chains and B_1 are one-boundaries. Another example is given in Fig. 14, where A, B, C are cycles and a boundary C, but A, B are not boundaries.

Topology

Conceptual consistency in scientific visualization is provided by topology. Cell complices convey topology in a computationally treatable manner and can therefore be introduced by much simpler definitions. A topological space $(X, \mathcal{T})$ is the collection of sets $\mathcal{T}$ that include:

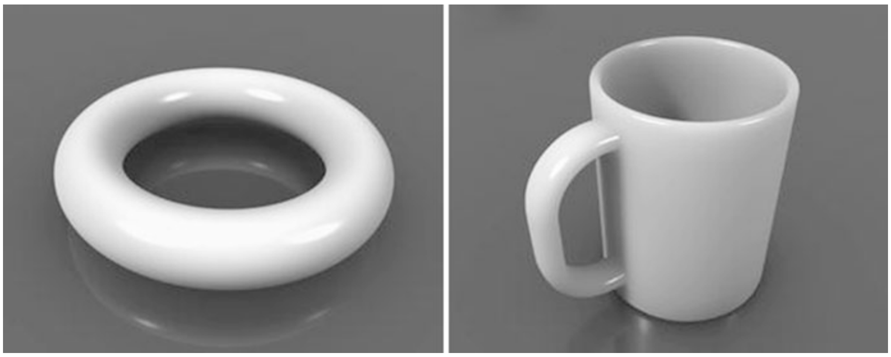


Fig. 15 Topologically a torus and a coffee mug are equivalent and so have the same Betti numbers

- The space itself X and the empty set $\emptyset$
- The union of any of these sets
- The finite intersection of any of the sets

The family $\mathcal{T}$ is called a *topology* on X, and the members of $\mathcal{T}$ are called *open sets*. As an example a basic set $X = \{a, b, c\}$ and a topology is given:

$$\begin{aligned}(X, \mathcal{T}) = \{\ & \emptyset,\\ & \{a\}, \{b\}, \{c\},\\ & \{a, b\}, \{a, c\}, \{b, c\},\\ & \{a, b, c\}\}\end{aligned}$$

The general definition for a topological space is very abstract and allows several topological spaces which are not useful in scientific visualization, e.g., a topological space $(X, \mathcal{T})$ with a trivial topology $\mathcal{T} = \{\emptyset, X\}$. So basic mechanisms of separation within a topological space are required, e.g., the Hausdorff property. A topological space $(X, \mathcal{T})$ is said to be Hausdorff if, given $x, y \in X$ with $x \neq y$, there exist open sets U_1, U_2 such that $x \in U_1, y \in U_2$ and $U_1 \cap U_2 = \emptyset$. But the question remains on what "topology" actually is. A brief explanation is given by the study of properties of an object that do not change under *deformation*. To describe this deformation process, abstract rules can be stated and if they are true, then an object A can be transformed into an object B without change. The two objects A, B are then called homeomorphic:

- All points of $A \leftrightarrow$ all points of B
- $1 - 1$ correspondence (no overlap)
- Bicontinous (continuous both ways)
- Cannot tear, join, poke/seal holes

The deformation is $1-1$ if each point of A maps to a single point on B and there is no overlap. If this deformation is continuous, A cannot be teared, joined, disrupted, or sealed up. If two objects are homeomorphic, then they are topologically equivalent. Figure 15 illustrates an example of a torus and coffee mug which are a prominent example for topological equivalence. The torus can be continuously deformed, without tearing, joining, disrupting, or sealing up, into a cup. The hole in the torus becomes the handle of the cup. But why should anybody in visualization be concerned about how objects can be deformed? Topology is much more than the illustrated properties, it can be much better described by the study of connectedness:

- Understanding of space properties: how connectivity happens.
- Analysis of space properties: how connectivity can be determined.
- Articulation of space properties: how connectivity can be described.
- Control about space properties: how connectivity can be enforced.

Topology studies properties of sets that do not change under well-behaved transformations (homeomorphisms). These properties include completeness and compactness. In visualization, one property is of significance: connectedness. Especially, how many disjoint components can be distinguished and how many holes (or tunnels) are in these components. Geometric configuration is another interesting aspect in visualization because it is important to know which of these components have how many holes, and where the holes are relative to each other. Several operations in scientific visualization can be summarized:

- Simplification: reduction of data complexity. If objects are described with fewer properties, important properties such as components or holes should be retained or removed, if these properties become insignificant, unnecessary, or imperceptible.
- Compression: reduction of data storage. It is important that each operation does not alter important features (interaction of geometrical and topological features).
- Texturing: visualization context elements. How can a texture kept consistent if an object, e.g., a torus, is transformed into another object, e.g., a coffee cup.
- Morphing: transforming one object into another. If an object is morphed into another, topological features have to remain, e.g., the torus hole has to become the coffee cup handle hole.

4 Geometric Algebra Computing

Geometric Algebra as a general mathematical system unites many mathematical concepts such as vector algebra, quaternions, Plücker coordinates, and projective geometry, and it easily deals with geometric objects, operations, and transformations. A lot of applications in computer graphics, computer vision, and other engineering areas can benefit from these properties. In a ray-tracing application, for

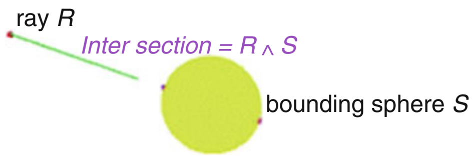


Fig. 16 Spheres and lines are basic entities of Geometric Algebra to compute with. Operations like the intersection of them are easily expressed with the help of their outer product. The result of the intersection of a ray and a (bounding) sphere is another geometric entity, the point pair of the two points of the line intersecting the sphere. The sign of the square of the point pair easily indicates whether there is a real intersection or not

instance, the intersection of a ray and a bounding sphere is needed. According to Fig. 16, this can be easily expressed with the help of the outer product of these two geometric entities.

Geometric Algebra is based on the work of Hermann Grassmann (see the conference [62] celebrating his 200th birthday in 2009) and William Clifford [20, 21]. Pioneering work has been done by David Hestenes, who first applied Geometric Algebra to problems in mechanics and physics [39, 40].

The first time Geometric Algebra was introduced to a wider computer graphics audience was through a couple of courses at the SIGGRAPH conferences in 2000 and 2001 (see [57]) and later at the Eurographics [41]. Researchers at the University of Cambridge, UK, have applied Geometric Algebra to a number of graphics-related projects. Geomerics [71] is a start-up company in Cambridge specializing in simulation software for physics and lighting, which presented its new technology allowing real-time radiosity in videogames utilizing commodity graphics-processing hardware. The technology is based on Geometric Algebra wavelet technology. Researchers at the University of Amsterdam, the Netherlands, are applying their fundamental research on Geometric Algebra to 3D computer vision and to ray tracing and on the efficient software implementation of Geometric Algebra. Researchers from Guadalajara, Mexico, are primarily dealing with the application of Geometric Algebra in the field of computer vision, robot vision, and kinematics. They are using Geometric Algebra, for instance, for tasks like visual-guided grasping, camera self-localization, and reconstruction of shape and motion. Their methods for geometric neural computing are used for tasks like pattern recognition [5]. Registration, the task of finding correspondences between two point sets, is solved based on Geometric Algebra methods in [65]. Some of their kinematics algorithms are dealing with inverse kinematics, fixation, and grasping as well as with kinematics and differential kinematics of binocular robot heads. At the University of Kiel, Germany, researchers are applying Geometric Algebra to robot vision and pose estimation [66]. They also do some interesting research dealing, for instance, with neural networks based on Geometric Algebra [14]. In addition to these examples, there are many other applications like Geometric Algebra Fourier transforms for the visualization and analysis of vector fields [24] or classification and clustering of spatial patterns with Geometric Algebra [63] showing the wide area

Table 1 Multiplication table of the 2D Geometric Algebra. This algebra consists of basic algebraic objects of grade (dimension) 0, the scalar; of grade 1, the two basis vectors e_1 and e_2; and of grade 2, the bi-vector $e_1 \wedge e_2$, which can be identified with the imaginary number i squaring to -1

	1	e_1	e_2	$e_1 \wedge e_2$
1	1	e_1	e_2	$e_1 \wedge e_2$
e_1	e_1	1	$e_1 \wedge e_2$	e_2
e_2	e_2	$-e_1 \wedge e_2$	1	$-e_1$
$e_1 \wedge e_2$	$e_1 \wedge e_2$	$-e_2$	e_1	-1

Table 2 List of the basic geometric primitives provided by the 5D conformal Geometric Algebra. The bold characters represent 3D entities (**x** is a 3D point, **n** is a 3D normal vector, and $\mathbf{x}^2$ is the scalar product of the 3D vector **x**). The two additional basis vectors e_0 and e_∞ represent the origin and infinity. Based on the outer product, *circles* and *lines* can be described as intersections of two spheres, respectively two planes. The parameter r represents the radius of the sphere and the parameter d the distance of the plane to the origin

Entity	Representation
Point	$P = \mathrm{x} + \frac{1}{2}\mathrm{x}^2 e_\infty + e_0$
Sphere	$S = P - \frac{1}{2}r^2 e_\infty$
Plane	$\pi = \mathbf{n} + d e_\infty$
Circle	$Z = S_1 \wedge S_2$
Line	$L = \pi_1 \wedge \pi_2$

of possibilities of advantageously using this mathematical system in engineering applications.

Benefits of Geometric Algebra

As follows, we highlight some of the properties of Geometric Algebra that make it advantageous for a lot of engineering applications.

Unification of Mathematical Systems

In the wide range of engineering applications, many different mathematical systems are currently used. One notable advantage of Geometric Algebra is that it subsumes mathematical systems like vector algebra, complex analysis, quaternions, or Plücker coordinates. Table 1, for instance, describes how complex numbers can be identified within the 2D Geometric Algebra. This algebra does not only contain the two basis vectors e_1 and e_2 but also basis elements of grade (dimension) 0 and 2 representing the scalar and imaginary part of complex numbers.

Other examples are Plücker coordinates based on the description of lines in conformal geometric algebra (see section "Conformal Geometric Algebra") or quaternions as to be identified in Fig. 19 with their imaginary units.

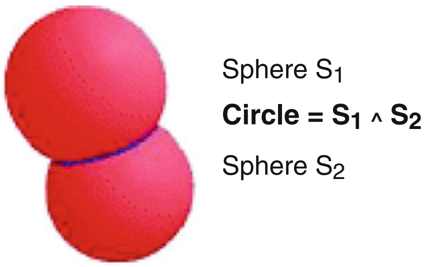


Fig. 17 Spheres and circles are basic entities of Geometric Algebra. Operations like the intersection of two spheres are easily expressed

Uniform Handling of Different Geometric Primitives

Conformal Geometric Algebra, the Geometric Algebra of conformal space we focus on, is able to easily treat different geometric objects. Table 2 presents the representation of points, lines, circles, spheres, and planes as the same entities algebraically. Consider the spheres of Fig. 17, for instance. These spheres are simply represented by

$$S = P - \frac{1}{2}r^2 e_\infty \tag{47}$$

based on their center point P, their radius r, and the basis vector e_∞ which represents the point at infinity. The circle of intersection of the spheres is then easily computed using the outer product to operate on the spheres as simply as if they were vectors:

$$Z = S_1 \wedge S_2 \tag{48}$$

This way of computing with Geometric Algebra clearly benefits computer graphics applications.

Simplified Geometric Operations

Geometric operations like rotations, translations (see [41]), and reflections can be easily treated within the algebra. There is no need to change the way of describing them with other approaches (vector algebra, for instance, additionally needs matrices in order to describe transformations).

Figure 18 visualizes the reflection of the ray R from one plane

$$\pi = \mathbf{n} + d e_\infty \tag{49}$$

(see Table 2). The reflected line, drawn in magenta,

$$\mathbf{R}_{\text{reflected}} = -\frac{\pi \mathbf{R}}{\pi} \tag{50}$$

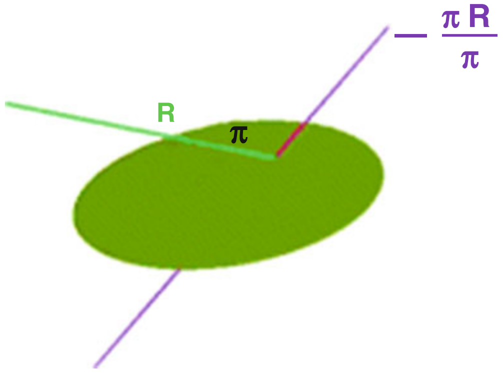


Fig. 18 The ray R is reflected from the plane π computing $-\frac{\pi\mathbf{R}}{\pi}$

is computed with the help of the reflection operation including the reflection object as well as the object to be reflected.

More Efficient Implementations

Geometric Algebra as a mathematical language suggests a clearer structure and greater elegance in understanding methods and formulae. But, what about the runtime performance for derived algorithms? Geometric Algebra inherently has a large potential for creating optimizations leading to more highly efficient implementations especially for parallel platforms. Gaalop [44], as presented in section "Computational Efficiency of Geometric Algebra Using Gaalop," is an approach offering dramatically improved optimizations.

Conformal Geometric Algebra

Conformal Geometric Algebra is a 5D Geometric Algebra based on the 3D basis vectors e_1, e_2, and e_3 as well as on the two additional base vectors e_0 representing the origin and e_∞ representing infinity.

Blades are the basic computational elements and the basic geometric entities of geometric algebras. The 5D conformal Geometric Algebra consists of blades with *grades* (dimension) 0, 1, 2, 3, 4, and 5, whereby a scalar is a *0-blade* (blade of grade 0). The element of grade five is called the pseudoscalar. A linear combination of blades is called a *k-vector*. So a bi-vector is a linear combination of blades with grade 2. Other k-vectors are vectors (grade 1), tri-vectors (grade 3), and quadvectors (grade 4). Furthermore, a linear combination of blades of different grades is called a *multivector*. Multivectors are the general elements of a Geometric Algebra. Table 3 lists all the 32 blades of conformal Geometric Algebra. The indices indicate 1, scalar; 2–6, vector; 7–16, bi-vector; 17–26, tri-vector; 27–31, quadvector; and 32, pseudoscalar.

A point $P = x_1e_1 + x_2e_2 + x_3e_3 + \frac{1}{2}\mathbf{x}^2e_\infty + e_0$ (see Table 2), for instance, can be written in terms of a multivector as the following linear combination of blades $b[i]$:

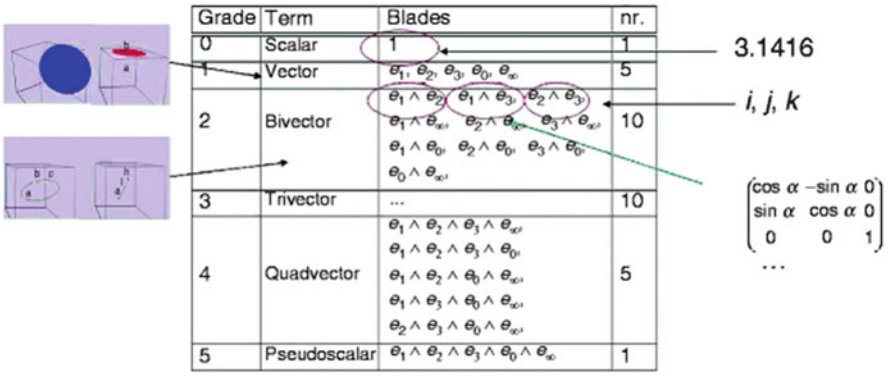
Table 3 The 32 blades of the 5D conformal Geometric Algebra

Index	Blade	Grade
1	1	0
2	e_1	1
3	e_2	1
4	e_3	1
5	e_∞	1
6	e_0	1
7	$e_1 \wedge e_2$	2
8	$e_1 \wedge e_3$	2
9	$e_1 \wedge e_\infty$	2
10	$e_1 \wedge e_0$	2
11	$e_2 \wedge e_3$	2
12	$e_2 \wedge e_\infty$	2
13	$e_2 \wedge e_0$	2
14	$e_3 \wedge e_\infty$	2
15	$e_3 \wedge e_0$	2
16	$e_\infty \wedge e_0$	2
17	$e_1 \wedge e_2 \wedge e_3$	3
18	$e_1 \wedge e_2 \wedge e_\infty$	3
19	$e_1 \wedge e_2 \wedge e_0$	3
20	$e_1 \wedge e_3 \wedge e_\infty$	3
21	$e_1 \wedge e_3 \wedge e_0$	3
22	$e_1 \wedge e_\infty \wedge e_0$	3
23	$e_2 \wedge e_3 \wedge e_\infty$	3
24	$e_2 \wedge e_3 \wedge e_0$	3
25	$e_2 \wedge e_\infty \wedge e_0$	3
26	$e_3 \wedge e_\infty \wedge e_0$	3
27	$e_1 \wedge e_2 \wedge e_3 \wedge e_\infty$	4
28	$e_1 \wedge e_2 \wedge e_3 \wedge e_0$	4
29	$e_1 \wedge e_2 \wedge e_\infty \wedge e_0$	4
30	$e_1 \wedge e_3 \wedge e_\infty \wedge e_0$	4
31	$e_2 \wedge e_3 \wedge e_\infty \wedge e_0$	4
32	$e_1 \wedge e_2 \wedge e_3 \wedge e_\infty \wedge e_0$	5

$$P = x_{1*}b[2] + x_{2*}b[3] + x_{3*}b[4] + \frac{1}{2}\mathbf{x}_*^2 b[5] + b[6] \tag{51}$$

with multivector indices according to Table 3.

Figure 19 describes some interpretations of the 32 basis blades of conformal Geometric Algebra. Scalars like the number π are grade 0 entities. They can be combined with the blade representing the imaginary unit i to complex numbers or with the blades representing the imaginary units i, j, k to quaternions. Since quaternions describe rotations, this kind of transformation can be handled within the



Grade	Term	Blades	nr.
0	Scalar	1	1
1	Vector	$e_1, e_2, e_3, e_0, e_\infty$	5
2	Bivector	$e_1 \wedge e_2, e_1 \wedge e_3, e_2 \wedge e_3,$ $e_1 \wedge e_\infty, e_2 \wedge e_\infty, e_3 \wedge e_\infty,$ $e_1 \wedge e_0, e_2 \wedge e_0, e_3 \wedge e_0,$ $e_0 \wedge e_\infty,$	10
3	Trivector	...	10
4	Quadvector	$e_1 \wedge e_2 \wedge e_3 \wedge e_\infty,$ $e_1 \wedge e_2 \wedge e_3 \wedge e_0,$ $e_1 \wedge e_2 \wedge e_0 \wedge e_\infty,$ $e_1 \wedge e_3 \wedge e_0 \wedge e_\infty,$ $e_2 \wedge e_3 \wedge e_0 \wedge e_\infty,$	5
5	Pseudoscalar	$e_1 \wedge e_2 \wedge e_3 \wedge e_0 \wedge e_\infty$	1

3.1416
i, j, k
$\begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$
...

Fig. 19 The blades of conformal Geometric Algebra. *Spheres* and *planes*, for instance, are vectors. *Lines* and *circles* can be represented as bi-vectors. Other mathematical systems like complex numbers or quaternions can be identified based on their imaginary units i, j, k. This is why also transformations like rotations can be handled within the algebra

Table 4 The extended list of the two representations of the conformal geometric entities. The IPNS representations as described in Table 1 have also an OPNS representation, which are dual to each other (indicated by the star symbol). In the OPNS representation, the geometric objects are described with the help of the outer product of conformal points that are part of the objects, for instance, lines as the outer product of two points and the point at infinity

Entity	IPNS representation	OPNS representation
Point	$P = \mathbf{x} + \frac{1}{2}\mathbf{x}^2 e_\infty + e_0$	
Sphere	$S = P - \frac{1}{2}r^2 e_\infty$	$S^* = P_1 \wedge P_2 \wedge P_3 \wedge P_4$
Plane	$\pi = \mathbf{n} + d e_\infty$	$\pi^* = P_1 \wedge P_2 \wedge P_3 \wedge e_\infty$
Circle	$Z = S_1 \wedge S_2$	$Z^* = P_1 \wedge P_2 \wedge P_3$
Line	$L = \pi_1 \wedge \pi_2$	$L^* = P_1 \wedge P_2 \wedge e_\infty$
Point pair	$Pp = S_1 \wedge S_2 \wedge S_3$	$Pp^* = P_1 \wedge P_2$

algebra. Geometric objects like spheres, planes, circles, and lines can be represented as vectors and bi-vectors.

Table 4 lists the two representations of the conformal geometric entities. The inner product null space (IPNS) and the outer product null space (OPNS) [61] are dual to each other. While Table 2 already presented the IPNS representation of spheres and planes, they can be described also with the outer product of four points being part of them. In the case of a plane one of these four points is the point at infinity e_∞. Circles can be described with the help of the outer product of three conformal points lying on the circle or as the intersection of two spheres.

Lines can be described with the help of the outer product of two points and the point at infinity e_∞ or with the help of the outer product of two planes (i.e., intersection in IPNS representation). An alternative expression is

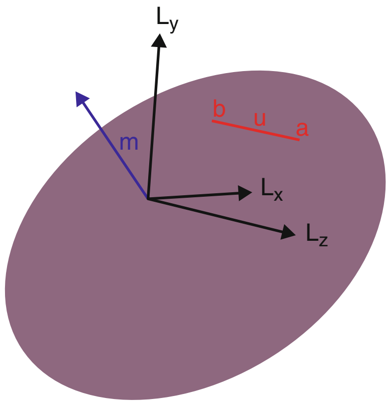


Fig. 20 The line L through the 3D points $\mathbf{a}$, $\mathbf{b}$ and the visualization of its 6D Plücker parameters based on the two 3D vectors $\mathbf{u}$ and $\mathbf{m}$ of Eq. (53)

$$L = \mathbf{u}e_{123} + \mathbf{m} \wedge e_\infty \tag{52}$$

with the 3D pseudoscalar $e_{123} = e_1 \wedge e_2 \wedge e_\infty$, the two 3D points $\mathbf{a}$, $\mathbf{b}$ on the line, $\mathbf{u} = \mathbf{b} - \mathbf{a}$ as 3D direction vector, and $\mathbf{m} = \mathbf{a} \times \mathbf{b}$ as the 3D moment vector (relative to origin). The corresponding six Plücker coordinates (components of $\mathbf{u}$ and $\mathbf{m}$) are (see Fig. 20)

$$(\mathbf{u} : \mathbf{m}) = (u_1 : u_2 : u_3 : m_1 : m_2 : m_3) \tag{53}$$

Computational Efficiency of Geometric Algebra Using Gaalop

Because of its generality, Geometric Algebra needs some optimizations for efficient implementations.

Gaigen [27] is a Geometric Algebra code generator developed at the University of Amsterdam (see [23, 26]). The philosophy behind Gaigen 2 is based on two ideas: generative programming and specializing for the structure of Geometric Algebra. Please find some benchmarks comparing Gaigen 2 with other pure software solutions as well as comparing five models of 3D Euclidean geometry for a ray-tracing application in [26, 28].

Gaalop [44] combines the advantages of software optimizations and the adaptability on different parallel platforms. As an example, an inverse kinematics algorithm of a computer animation application was investigated [42]. With the optimization approach of Gaalop, the software implementation became three times faster and with a hardware implementation about 300 times faster [43] (three times by software optimization and 100 times by additional hardware optimization) than the conventional software implementation. Figure 21 shows an overview over the architecture of Gaalop. Its input is a Geometric Algebra algorithm written in CLUCalc [60], a system for the visual development of Geometric Algebra

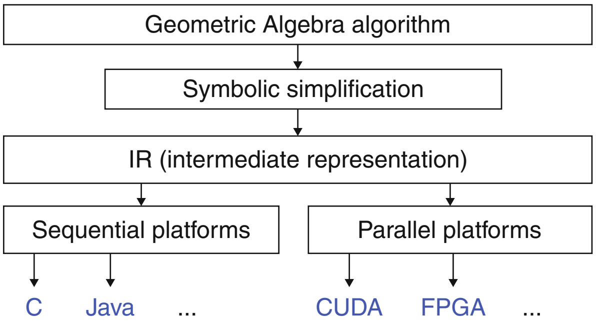


Fig. 21 Architecture of Gaalop

algorithms. Via symbolic simplification it is transformed into an intermediate representation (IR) that can be used for the generation of different output formats. Gaalop supports sequential platforms with the automatic generation of C and JAVA code while its main focus is on supporting parallel platforms like reconfigurable hardware as well as modern accelerating GPUs.

Gaalop uses the symbolic computation functionality of Maple (using the Open Maple interface and a library for Geometric Algebras [1]) in order to optimize a Geometric Algebra algorithm. It computes the coefficients of the desired multivector symbolically, returning an efficient implementation depending just on the input variables.

As an example, the following CLUCalc code computes the intersection circle C of two spheres $S1$ and $S2$ according to Fig. 17:

```
P1 = x1*e1 + x2*e2 + x3*e3 + 1/2*(x1*x1+x2*x2+x3*x3)*einf
     + e0; P2 = y1*e1 +
y2*e2 +y3*e3 + 1/2*(y1*y1+y2*y2+y3*y3)*einf + e0;
     S1 =P1 - 1/2 * r1*r1 *
einf; S2 = P2 - 1/2 * r2*r2 * einf; ?C = S1 $\wedge$ S2;
```

See Table 2 for the computation of the conformal points $P1$ and $P2$, the spheres $S1$ and $S2$, as well as the resulting circle based on the outer product of the two spheres.

The resulting C code generated by Gaalop for the intersection circle C is as follows and depends only on the variables $x1$, $x2$, $x3$, $y1$, $y2$, $y3$, $r1$, and $r2$ for the 3D center points and radii:

```
float C [32] = {\{}0.0{\}}; C[7] = x1*y2-x2*y1;
   C[8] = x1*y3-x3*y1; C[9]
= -0.5*y1*x1*x1-0.5*y1*x2*x2 -0.5*y1*x3*x3+0.5*y1*r1*r1 +
0.5*x1*y1*y1+0.5*x1*y2*y2 + 0.5*x1*y3*y3 - 0.5*x1*r2*r2;
   C[10] = -y1 +
x1; C[11] = -x3*y2+x2*y3; C[12] = -0.5*y2*x1*x1-0.5*y2*x2*x2-
0.5*y2*x3*x3 + 0.5*y2*r1*r1 + 0.5*x2*y1*y1 + 0.5*x2*y2*y2 +
   0.5*x2*y3*y3 -
0.5*x2*r2*r2; C[13] = -y2 + x2; C[14] = -0.5*y3*x1*x1 -
   0.5*y3*x2*x2
```

```
-0.5*y3*x3*x3 + 0.5*y3*r1*r1 + 0.5*x3*y1*y1 + 0.5*x3*y2*y2 +
   0.5*x3*y3*y3
- 0.5*x3*r2*r2; C[15] = -y3 + x3; C[16] = -0.5*y3*y3 +
   0.5*x3*x3 +
0.5*x2*x2 + 0.5*r2*r2 -0.5*y1*y1 - 0.5*y2*y2 + 0.5*x1*x1 -
   0.5*r1*r1;
```

In a nutshell, Gaalop always computes optimized 32-dimensional multivectors. Since a circle is described with the help of a bi-vector, only the blades 7–16 (see Table 3) are used. As you can see, all the corresponding coefficients of this multivector are very simple expressions with basic arithmetic operations.

5 Feature-Based Vector Field Visualization

We will identify derived quantities that describe flow features such as vortices (section "Derived Measures of Vector Fields") and we discuss the topology of vector fields (section "Topology of Vector Fields"). However, not all feature-based visualization approaches can be covered here. The reader is referred to [84] for further information on this topic. We start with a description of integral curves in vector fields, which are the basis for most feature-based visualization approaches.

Characteristic Curves of Vector Fields

A curve $q : \mathbb{R} \to M$ (see section "Tangential Vectors") is called a *tangent curve* of a vector field $\mathbf{v}(\mathbf{x})$, if for all points $\mathbf{x} \in q$ the tangent vector $\dot{q}$ of q coincides with $\mathbf{v}(\mathbf{x})$. Tangent curves are the solutions of the autonomous ODE system

$$\frac{d}{d\tau}\mathbf{x}(\tau) = \mathbf{v}(\mathbf{x}(\tau)) \quad \text{with } \mathbf{x}(0) = \mathbf{x}_0 \tag{54}$$

For all points $\mathbf{x} \in M$ with $\mathbf{v}(\mathbf{x}) \neq \mathbf{0}$, there is one and only one tangent curve through it. Tangent curves do not intersect or join each other. Hence, tangent curves uniquely describe the directional information and are therefore an important tool for visualizing vector fields.

The tangent curves of a parameter-independent vector field $\mathbf{v}(\mathbf{x})$ are called *streamlines*. A streamline describes the path of a massless particle in $\mathbf{v}$.

In a one-parameter-dependent vector field $\mathbf{v}(\mathbf{x}, t)$, there are four types of characteristic curves: streamlines, path lines, streak lines, and time lines. To ease the explanation, we consider $\mathbf{v}(\mathbf{x}, t)$ as a time-dependent vector field in the following: in a space-time point $(\mathbf{x}_0, t_0)$ we can start a *streamline* (staying in time slice $t = t_0$) by integrating

$$\frac{d}{d\tau}\mathbf{x}(\tau) = \mathbf{v}(\mathbf{x}(\tau), t_0) \quad \text{with } \mathbf{x}(0) = \mathbf{x}_0 \tag{55}$$

or a *path line* by integrating

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{v}(\mathbf{x}(t), t) \quad \text{with } \mathbf{x}(t_0) = \mathbf{x}_0 \tag{56}$$

Path lines describe the trajectories of massless particles in time-dependent vector fields. The ODE system (56) can be rewritten as an autonomous system at the expense of an increase in dimension by one, if time is included as an explicit state variable:

$$\frac{d}{dt}\begin{pmatrix}\mathbf{x}\\ t\end{pmatrix} = \begin{pmatrix}\mathbf{v}(\mathbf{x}(t), t)\\ 1\end{pmatrix} \quad \text{with } \begin{pmatrix}\mathbf{x}\\ t\end{pmatrix}(0) = \begin{pmatrix}\mathbf{x}_0\\ t_0\end{pmatrix} \tag{57}$$

In this formulation space and time are dealt with on equal footing – facilitating the analysis of spatiotemporal features. Path lines of the original vector field **v** in ordinary space now appear as tangent curves of the vector field

$$\mathbf{p}(\mathbf{x}, t) = \begin{pmatrix}\mathbf{v}(\mathbf{x}, t)\\ 1\end{pmatrix} \tag{58}$$

in space-time. To treat streamlines of **v**, one may simply use

$$\mathbf{s}(\mathbf{x}, t) = \begin{pmatrix}\mathbf{v}(\mathbf{x}, t)\\ 0\end{pmatrix} \tag{59}$$

Figure 22 illustrates **s** and **p** for a simple example vector field **v**. It is obtained by a linear interpolation over time of two bilinear vector fields. Figure 22a depicts streamlines, Fig. 22b depicts pathlines.

A *streak line* is the connection of all particles set out at different times but the same point location. In an experiment, one can observe these structures by constantly releasing dye into the flow from a fixed position. The resulting streak line consists of all particles which have been at this fixed position sometime in the past. Considering the vector field **p** introduced above, streak lines can be obtained in the following way: apply a stream surface integration in **p** where the seeding curve is a straight line segment parallel to the t-axis; a streak line is the intersection of this stream surface with a hyperplane perpendicular to the t-axis (Fig. 22c).

A *time line* is the connection of all particles set out at the same time but different locations, i.e., a line which gets advected by the flow. An analogon in the real world is a yarn or wire thrown into a river, which gets transported and deformed by the flow. However, in contrast to the yarn, a time line can get shorter and longer. It can be obtained by applying a stream surface integration in **p** starting at a line with $t = \{\text{const.}\}$ and intersecting it with a hyperplane perpendicular to the t-axis (Fig. 22d).

Streak lines and time lines cannot be described as tangent curves in the spatiotemporal domain. Both types of lines fail to have a property of stream and path lines: they are not locally unique, i.e., for a particular location and time there

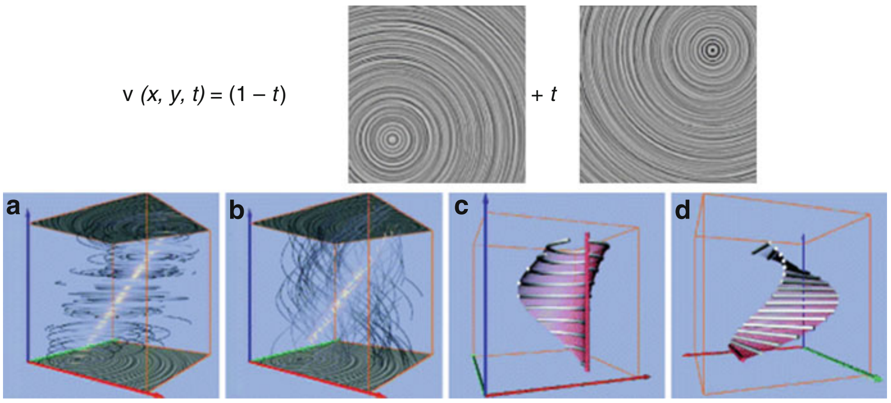


Fig. 22 Characteristic curves of a simple 2D time-dependent vector field. Streamlines and path lines are shown as illuminated field lines. Streak and time lines are shown as thick cylindrical lines, while their seeding curves and resulting stream surfaces are colored *red*. The *red/green* coordinate axes denote the (x, y)-domain; the *blue axis* shows time

is more than one streak and time line passing through. However, stream, path, and streak lines coincide for steady vector fields $\mathbf{v}(\mathbf{x}, t) = \mathbf{v}(\mathbf{x}, t_0)$ and are described by (54) in this setting. Time lines do not fit into this.

Derived Measures of Vector Fields

A number of measures can be derived from a vector field **v** and its derivatives. These measures indicate certain properties or features and can be helpful when visualizing flows. The following text assumes the underlying manifold M where the vector field is given to be Euclidean space, i.e., the manifold is three-dimensional and Cartesian coordinates are used where the metric (see section "Tensors") is representable as the unit matrix.

The *magnitude* of **v** is then given as

$$|\mathbf{v}| = \sqrt{u^2 + v^2 + w^2} \tag{60}$$

The *divergence* of a flow field is given as

$$\operatorname{div}(\mathbf{v}) = \nabla \cdot \mathbf{v} = \operatorname{trace}(\mathbf{J}) = u_x + v_y + w_z \tag{61}$$

and denotes the gain or loss of mass density at a certain point of the vector field: given a volume element in a flow, a certain amount of mass is entering and exiting it. Divergence is the net flux of this at the limit of a point. A flow field with $\{\operatorname{div}\}(\mathbf{v}) = 0$ is called *divergence-free*, which is a common case in fluid dynamics since a number of fluids are *incompressible*.

The *vorticity* or *curl* of a flow field is given as

$$\omega = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix} = \nabla \times \mathbf{v} = \begin{pmatrix} w_y - v_z \\ u_z - w_x \\ v_x - u_y \end{pmatrix} \tag{62}$$

This vector is the axis of locally strongest rotation, i.e., it is perpendicular to the plane in which the locally highest amount of circulation takes place. The vorticity magnitude $|\omega|$ gives the strength of rotation and is often used to identify regions of high vortical activity. A vector field with $\omega = \mathbf{0}$ is called *irrotational* or *curl-free*, with the important subclass of *conservative* vector fields, i.e., vector fields which are the gradient of a scalar field. Note that Geometric Algebra (see section "Geometric Algebra" and Sect. 4) treats Eqs. (61) and (62) as an entity, called the geometric derivative. The identification of vortices is a major subject in fluid dynamics. The most widely used quantities for detecting vortices are based on a decomposition of the Jacobian matrix $\mathbf{J} = \mathbf{S} + \Omega$ into its symmetric part, the strain tensor

$$\mathbf{S} = \frac{1}{2}(\mathbf{J} + \mathbf{J}^T) \tag{63}$$

and its antisymmetric part, the vorticity tensor

$$\Omega = \frac{1}{2}(\mathbf{J} - \mathbf{J}^T) = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \tag{64}$$

with ω_i being the components of vorticity (62). While Ω assesses vortical activity, the strain tensor $\mathbf{S}$ measures the amount of stretching and folding which drives mixing to occur.

Inherent to the decomposition of the flow field gradient $\mathbf{J}$ into $\mathbf{S}$ and Ω is the following duality: vortical activity is high in regions where Ω dominates $\mathbf{S}$, whereas strain is characterized by $\mathbf{S}$ dominating Ω.

In order to identify vortical activity, Jeong et al. used this decomposition in [47] to derive the vortex region quantity λ_2 as the second largest eigenvalue of the symmetric tensor $\mathbf{S}^2 + \Omega^2$. Vortex regions are identified by $\lambda_2 < 0$, whereas $\lambda_2 > 0$ lacks physical interpretation. λ_2 does not capture stretching and folding of fluid particles and hence does not capture the vorticity-strain duality.

The Q-criterion of Hunt [46], also known as the Okubo-Weiss criterion, is defined by

$$Q = \frac{1}{2}(\| \Omega \|^2 - \| \mathbf{S} \|^2) = \| \omega \|^2 - \frac{1}{2} \| \mathbf{S} \|^2 \tag{65}$$

where Q is positive and the vorticity magnitude dominates the rate of strain. Hence it is natural to define vortex regions as regions where $Q > 0$. Unlike λ_2, Q has a physical meaning also where $Q < 0$. Here the rate of strain dominates the vorticity magnitude.

Topology of Vector Fields

In this section we collect the first-order topological properties of steady 2D and 3D vector fields. The extraction of these topological structures has become a standard tool in visualization for the feature-based analysis of vector fields.

Critical Points

Considering a steady vector field $\mathbf{v}(\mathbf{x})$, an isolated *critical point* $\mathbf{x}_0$ is given by

$$\mathbf{v}(\mathbf{x}_0) = \mathbf{0} \quad \text{with} \quad \mathbf{v}(\mathbf{x}_0 \pm \epsilon) \neq \mathbf{0} \tag{66}$$

This means that $\mathbf{v}$ is zero at the critical point but nonzero in a certain neighborhood.

Every critical point can be assigned an *index*. For a 2D vector field it denotes the number of counterclockwise revolutions of the vectors of $\mathbf{v}$ while traveling counterclockwise on a closed curve around the critical point (for 2D vector fields, it is therefore often called the *winding number*). Similarly, the index of a 3D critical point measures the number of times the vectors of $\mathbf{v}$ cover the area of an enclosing sphere. The index is always an integer and it may be positive or negative. For a curve/sphere enclosing an arbitrary part of a vector field, the index of the enclosed area/volume is the sum of the indices of the enclosed critical points. Mann et al. show in [54] how to compute the index of a region using Geometric Algebra. A detailed discussion of index theory can be found in [25, 31, 32].

Critical points are characterized and classified by the behavior of the tangent curves around it. Here we concentrate on first-order critical points, i.e., critical points with $\det(\mathbf{J}(\mathbf{x}_0)) \neq 0$. As shown in [37, 38], a first-order Taylor expansion of the flow around $\mathbf{x}_0$ suffices to completely classify it. This is done by an eigenvalue/eigenvector analysis of $\mathbf{J}(\mathbf{x}_0)$. Let λ_i be the eigenvalues of $\mathbf{J}(\mathbf{x}_0)$ ordered according to their real parts, i.e., $\text{Re}(\lambda_{i-1}) \leq \text{Re}(\lambda_i)$. Furthermore, let $\mathbf{e}_i$ be the corresponding eigenvectors, and let $\mathbf{f}_i$ be the corresponding eigenvectors of the transposed Jacobian $(\mathbf{J}(\mathbf{x}_0))^T$ (note that $\mathbf{J}$ and $\mathbf{J}^T$ have the same eigenvalues but not necessarily the same eigenvectors). The sign of the real part of an eigenvalue λ_i denotes – together with the corresponding eigenvector $\mathbf{e}_i$ – the flow direction: positive values represent an *outflow* and negative values an *inflow* behavior. Based on this we give the classification of 2D and 3D first-order critical points in the following.

2D Vector Fields Based on the flow direction, first-order critical points in 2D vector fields are classified into:

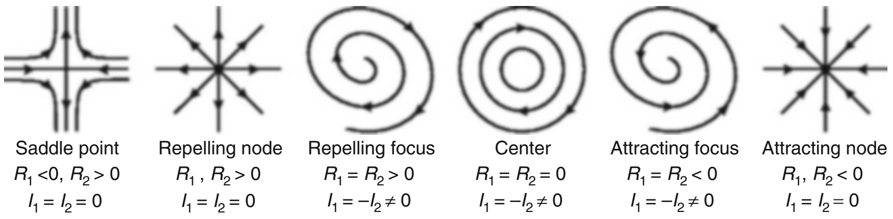


Fig. 23 Classification of first-order critical points. R_1, R_2 denote the real parts of the eigenvalues of the Jacobian matrix, while I_1, I_2 denote their imaginary parts (From [37])

$$\begin{aligned} &\text{Sources}: & & 0 < \operatorname{Re}(\lambda_1) \leq \operatorname{Re}(\lambda_2) \\ &\text{Saddles}: \operatorname{Re}(\lambda_1) < & & 0 < \operatorname{Re}(\lambda_2) \\ \text{Sinks}: \operatorname{Re}(\lambda_1) \leq & \operatorname{Re}(\lambda_2) < & & 0 \end{aligned}$$

Thus, sources and sinks consist of complete outflow/inflow, while saddles have a mixture of both.

Sources and sinks can be further divided into two stable subclasses by deciding whether or not imaginary parts are present, i.e., whether or not λ_1, λ_2 is a pair of conjugate complex eigenvalues:

$$\begin{aligned} &\text{Foci}: \quad \operatorname{Im}(\lambda_1) = -\operatorname{Im}(\lambda_2) \neq 0 \\ &\text{Nodes}: \qquad\qquad \operatorname{Im}(\lambda_1) = \operatorname{Im}(\lambda_2) = 0 \end{aligned}$$

There is another important class of critical points in 2D: a *center*. Here, we have a pair of conjugate complex eigenvalues with $\operatorname{Re}(\lambda_1) = \operatorname{Re}(\lambda_2) = 0$. This type is common in incompressible (divergence-free) flows but unstable in general vector fields since a small perturbation of $\mathbf{v}$ changes the center to either a sink or a source. Figure 23 shows the phase portraits of the different types of first-order critical points following [37].

The index of a saddle point is -1, while the index of a source, sink, or center is $+1$. It turns out that this coincides with the sign of $\det(\mathbf{J}(\mathbf{x}_0))$: a negative determinant denotes a saddle, a positive determinant a source, sink, or center. This already shows that the index of a critical point cannot be used to distinguish or classify them completely, since different types like sources and sinks have assigned the same index.

An iconic representation is an appropriate visualization for critical points, since vector fields usually contain a finite number of them. We will display them as spheres colored according to their classification: sources will be colored in red, sinks in blue, saddles in yellow, and centers in green.

3D Vector Fields Depending on the sign of $\operatorname{Re}(\lambda_i)$ we get the following classification of first-order critical points in 3D vector fields:

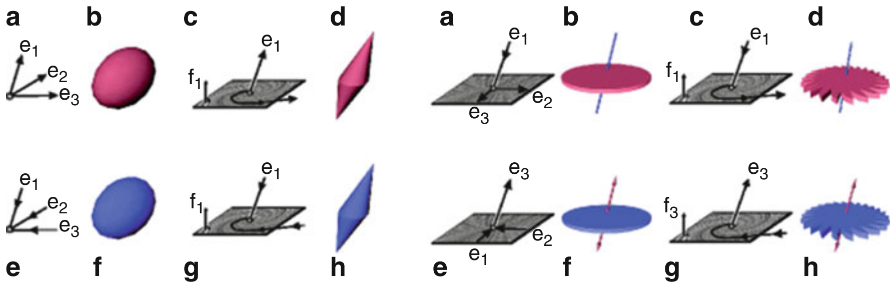


Fig. 24 Flow behavior around critical points of 3D vector fields and corresponding iconic representation

$$\begin{array}{rllll}
 & \text{Sources :} & & 0 & < \\
\text{Repelling saddles :} & \mathrm{Re}(\lambda_1) < 0 & & < \mathrm{Re}(\lambda_2) \le & \mathrm{Re}(\lambda_3) \\
\text{Attracting saddles :} & \mathrm{Re}(\lambda_1) \le & \mathrm{Re}(\lambda_2) < 0 & < & \mathrm{Re}(\lambda_3) \\
 & \text{Sinks :} & & \mathrm{Re}(\lambda_1) \le & \mathrm{Re}(\lambda_2) \le
\end{array}$$

Again, sources and sinks consist of complete outflow/inflow, while saddles have a mixture of both. A repelling saddle has one direction of inflow behavior (called *inflow direction*) and a plane in which a 2D outflow behavior occurs (called *outflow plane*). Similar to this, an attracting saddle consists of an *outflow direction* and an *inflow plane*.

Each of the four classes above can be further divided into two stable subclasses by deciding whether or not imaginary parts in two of the eigenvalues are present ($\lambda_1, \lambda_2, \lambda_3$ are not ordered):

$$\begin{array}{ll}
\text{Foci :} & \mathrm{Im}(\lambda_1) = 0 \quad \text{and} \quad \mathrm{Im}(\lambda_2) = -\mathrm{Im}(\lambda_3) \neq 0 \\
\text{Nodes :} & \mathrm{Im}(\lambda_1) = \mathrm{Im}(\lambda_2) = \mathrm{Im}(\lambda_3) = 0
\end{array}$$

As argued in [29], the index of a first-order critical point is given as the sign of the product of the eigenvalues of $\mathbf{J}(\mathbf{x}_0)$. This yields an index of $+1$ for sources and attracting saddles and an index of -1 for sinks and repelling saddles.

In order to depict 3D critical points, several icons have been proposed in the literature; see [30, 36, 37, 53]. Basically, we follow the design approach of [72, 85] and color the icons depending on the flow behavior: attracting parts (inflow) are colored blue, while repelling parts (outflow) are colored red (Fig. 24).

Separatrices

Separatrices are streamlines or stream surfaces which separate regions of different flow behavior. Here we concentrate on separatrices that emanate from critical points. Due to the homogeneous flow behavior around sources and sinks (either a complete outflow or inflow), they do not contribute to separatrices. Each saddle point creates two separatrices: one in forward and one in backward integration into the directions

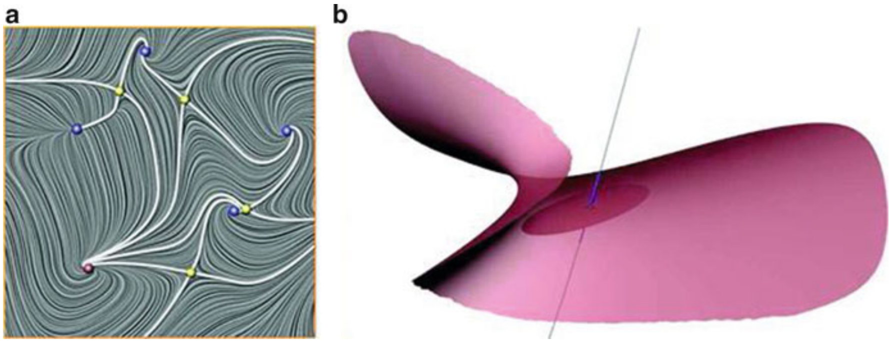


Fig. 25 Separatrices are streamlines or surfaces starting from saddle points into the direction of the eigenvectors

of the eigenvectors. For a 2D saddle point, this gives two separation lines (Fig. 25a). Considering a repelling saddle $\mathbf{x}_R$ of a 3D vector field, it creates one separation curve (which is a streamline starting in $\mathbf{x}_R$ in the inflow direction by backward integration) and a separation surface (which is a stream surface starting in the outflow plane by forward integration). Figure 25b gives an illustration. A similar statement holds for attracting saddles.

Other kinds of separatrices are possible as well: they can emanate from boundary switch curves [85] and attachment and detachment lines [48], or they are closed separatrices without a specific emanating structure [73].

Application

In the following, we exemplify the topological concepts described above by applying them to a 3D vector field. First, we extract the critical points by searching for zeros in the vector field. Based on an eigenvalue/eigenvector analysis, we identify the different types of the critical points. Starting from the saddles, we integrate the separatrices into the directions of the eigenvectors.

Figure 26 visualizes the electrostatic field around a benzene molecule. This data set was calculated on a 101^3 regular grid using the fractional charges method described in [70]. It consists of 184 first-order critical points depicted in Fig. 26a. The separation surfaces shown in Fig. 26b emanate from 78 attracting and 43 repelling saddles. Note how they hide each other as well as the critical points. Even rendering the surfaces in a semitransparent style does not reduce the visual clutter to an acceptable degree. This is one of the major challenges for the topological visualization of 3D vector fields.

Figure 26c shows a possible solution to this problem by showing the 129 *saddle connectors* that we found in this data set. Saddle connectors are the intersection curves of repelling and attracting separation surfaces and have been introduced to the visualization community in [72]. Despite the fact that saddle connectors can only indicate the approximate run of the separation surfaces, the resulting visualization gives more insight into the symmetry and three-dimensionality of the

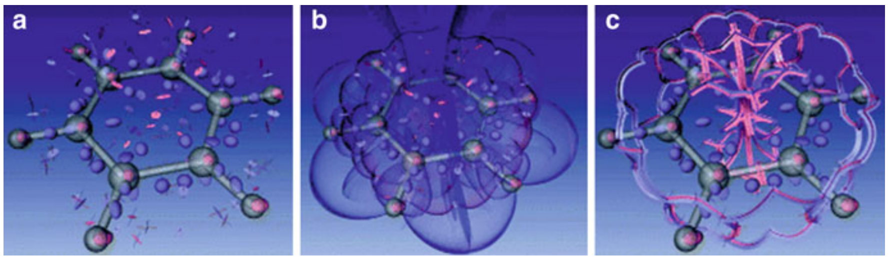


Fig. 26 Topological representations of the benzene data set with 184 critical points. (**a**) Iconic representation. (**b**) Due to the shown separation surfaces, the topological skeleton of the vector field looks visually cluttered. (**c**) Visualization of the topological skeleton using saddle connectors

data set. Saddle connectors are a useful compromise between the amount of coded information and the expressiveness of the visualization for complex topological skeletons.

6 Anisotropic Diffusion PDEs for Image Regularization and Visualization

Regularization PDEs: A Review

We consider a 2D multivalued image **I**: $\Omega \rightarrow \mathbb{R}^n (n = 3$ for color images) defined on a domain $\Omega \subset \mathbb{R}^2$ and denote by $I_i : \Omega \rightarrow \mathbb{R}$ the scalar channel i of **I**:

$$\forall \mathbf{X} = (x, y) \in \Omega, \mathbf{I}_{(\mathbf{X})} = (I_{1(\mathbf{X})} \ \ I_{2(\mathbf{X})} \ \ \ldots \ \ I_{n(\mathbf{X})})^T.$$

Local Multivalued Geometry and Diffusion Tensors

PDE-based regularization can be often seen as the local smoothing of an image **I** along defined directions depending themselves on the local configuration of the pixel intensities, i.e., one wants basically to smooth **I** in parallel to the image discontinuities. Naturally, this means that one has first to retrieve the *local geometry* of the image **I**. It consists in the definition of these important features at each image point $\mathbf{X} = (x, y) \in \Omega$:

- Two orthogonal directions $\theta_{(\mathbf{X})}{}^+, \ \theta_{(\mathbf{X})}{}^- \in \mathrm{S}^1$ along the local maximum and minimum variations of image intensities at **X**. θ^- is then considered to be parallel to the local edge, when there is one.
- Two corresponding positive values $\lambda_{(\mathbf{x})}{}^+, \lambda_{(\mathbf{x})}{}^-$ measuring the effective variations of the image intensities along $\theta_{(\mathbf{x})}{}^+$ and $\theta_{(\mathbf{x})}{}^-$, respectively. λ^-, λ^+ are related to the local *contrast* of an edge.

For scalar images $I : \Omega \rightarrow \mathbb{R}$, this local geometry $\{\lambda^{+/-}, \theta^{+/-} | \mathbf{X} \in \Omega\}$ is usually retrieved by the computation of the smoothed gradient field $\nabla \quad I_\sigma =$

$\nabla I * G_\sigma$ where G_σ is a 2D Gaussian kernel with standard deviation σ. Then, $\lambda^+ = \| \nabla I_\sigma \|^2$ is a possible measure of the local contrast of the contours, while $\theta^- = \nabla I_\sigma{}^\perp / \|\nabla I_\sigma\|$ gives the contours direction. Such a local geometry $\{\lambda^{+/-}, \theta^{+/-} | \mathbf{X} \in \Omega\}$ can be represented in a more convenient form by a field $\mathbf{G}$: $\Omega \to$ P(2) of second-order tensors (2×2 symmetric and semi-positive matrices):

$$\forall \mathbf{X} \in \Omega, \quad \mathbf{G}_{(\mathbf{X})} = \lambda^- \, \theta^- \theta^{-^T} + \lambda^+ \, \theta^+ \theta^{+^T}.$$

Eigenvalues of $\mathbf{G}$ are indeed λ^- and λ^+ and corresponding eigenvectors are θ^- and θ^+. The local geometry of scalar-valued images I can be then modeled by the tensor field $\mathbf{G}_{(\mathbf{x})} = \nabla I_{\sigma(\mathbf{x})} \nabla I^T_{\sigma(\mathbf{x})}$.

For multivalued images $\mathbf{I} : \Omega \to \mathbb{R}^n$, the local geometry can be retrieved in a similar way, by the computation of the field $\mathbf{G}$ of the smoothed *structure tensors*. As explained in [22, 82], this is a nice extension of the gradient for multivalued images:

$$\forall \mathbf{X} \in \Omega, \quad \mathbf{G}_{\sigma(\mathbf{X})} = \left(\sum_{i=1}^{n} \nabla I_{i\alpha(\mathbf{X})} \nabla I^T_{i\alpha(\mathbf{X})} \right)_* G_\sigma \quad \text{where} \quad \nabla I_{i\alpha} = \begin{pmatrix} \frac{\partial I_i}{\partial x} \\ \frac{\partial I_i}{\partial y} \end{pmatrix} * G_\alpha \tag{67}$$

$\mathbf{G}_{\sigma(\mathbf{x})}$ is a very good estimator of the local multivalued geometry of $\mathbf{I}$ at $\mathbf{X}$: its spectral elements give at the same time the vector-valued variations (by the eigenvalues λ^-, λ^+ of $\mathbf{G}_\sigma$) and the orientations (edges) of the local image structures (by the eigenvectors $\theta^- \perp \theta^+$ of $\mathbf{G}_\sigma$), σ being proportional to the so-called noise scale.

Once the local geometry $\mathbf{G}_\sigma$ of $\mathbf{I}$ has been determined, the way the regularization process is achieved is defined by another field $\mathbf{T}$: $\Omega \to$ P(2) of *diffusion tensors*, which specifies the local smoothing geometry that should drive the PDE flow. Of course, $\mathbf{T}$ depends on the targeted application, and most of the time it is constructed from the local geometry $\mathbf{G}_\sigma$ of $\mathbf{I}$. It is thus defined from the spectral elements λ^-, λ^+ and θ^-, θ^+ of $\mathbf{G}_\sigma$. In [19, 78], the following expression is proposed for image regularization:

$$\forall \mathbf{X} \in \Omega, \quad \mathbf{T}_{(\mathbf{X})} = f^-_{(\lambda^+,\lambda^-)} \, \theta^- \theta^- + f^+_{(\lambda^+,\lambda^-)} \, \theta^+ \theta^{+^T} \tag{68}$$

where

$$f^-_{(\lambda_+,\lambda_-)} = \frac{1}{(1 + \lambda^+ + \lambda^-)^{p_1}} \quad \text{and} \quad f^+_{(\lambda_+,\lambda_-)} = \frac{1}{(1 + \lambda^+ + \lambda^-)^{p_2}} \quad \text{with } p_1 < p_2$$

are the two functions which set the strengths of the desired smoothing along the respective directions θ^-, θ^+. This latest choice basically says that if a pixel $\mathbf{X}$ is located on an image contour ($\lambda^+_{(\mathbf{x})}$ is high), the smoothing on $\mathbf{X}$ would be performed mostly along the contour direction $\theta^-_{(\mathbf{x})}$ (since $f^+_{(.,.)} \ll f^-_{(.,.)}$). Conversely, if a pixel $\mathbf{X}$ is located on a homogeneous region ($\lambda^+_{(\mathbf{x})}$ is low), the smoothing on $\mathbf{X}$ would be performed in all possible directions (isotropic smoothing), since $f^+_{(\ldots)} \simeq f^-_{(\ldots)}$ (and then $\mathbf{T} \simeq \mathbb{I}_d$). Predefining the smoothing geometry $\mathbf{T}$ of each applied PDE iteration is the first stage of most of the PDE-based regularization algorithms. Most of the differences between existing regularization methods (as in [2,3,10,18,19,49,52,58, 59,67–69]) lie first on the definition of $\mathbf{T}$, but also on the kind of the diffusion PDE that will be used indeed to perform the desired smoothing.

Divergence-Based PDEs

One of the common choices to smooth a corrupted multivalued image $\mathbf{I}:\Omega \to \mathbb{R}^n$ following a local smoothing geometry $\mathbf{T}$: $\Omega \to$ P(2) is to use the divergence PDE:

$$\forall i = 1, \ldots, n, \quad \frac{\partial I_i}{\partial t} = \operatorname{div}(\mathbf{T}\nabla I_i) \tag{69}$$

The general form of this now classical PDE for image regularization has been introduced by Weickert in [82] and adapted for color/multivalued images in [83]. In this latter case, the tensor field $\mathbf{T}$ is chosen the same for all image channels I_i, ensuring that channels are smoothed with a *coherent multivalued geometry* which takes the correlation between channels into account (since $\mathbf{T}$ depends on $\mathbf{G}$). Equation (69) unifies a lot of existing scalar or multivalued regularization approaches and proposes at the same time two interpretation levels of the regularization process:

- *Local interpretation*: Equation (69) may be seen as the physical law describing local diffusion processes of the pixels individually regarded as temperatures or chemical concentrations in an anisotropic environment which is locally described by $\mathbf{T}$.
- *Global interpretation*: The problem of image regularization can be regarded as the minimization of the energy functional $E(\mathbf{I})$ by a gradient descent (i.e., a PDE), coming from the Euler-Lagrange equations of $E(\mathbf{I})$ [3,19,49,51,78]:

$$E(\mathbf{I}) = \int_\Omega \psi(\lambda^+, \lambda^-)\, d\Omega \qquad \text{where } \psi \;:\; \mathbb{R}^2 \to \mathbb{R} \tag{70}$$

- It results in a particular case of the PDE (69), with $\mathbf{T} = \frac{\partial\Psi}{\partial\lambda^-}\theta^-\theta^{-^T} + \frac{\partial\Psi}{\partial\lambda^+}\theta^+\theta^{+^T}$, where λ_+, λ_- are the two positive eigenvalues of the *non-smoothed* structure tensor field $\mathbf{G} = \sum_i \nabla I_i \; \nabla I_i^T$ and θ_+, θ_- are the corresponding eigenvectors.

Unfortunately, there are local configurations where the PDE (69) *does not fully respect the geometry* $\mathbf{T}$ and where the smoothing is performed in unexpected directions. For instance, considering (69) with tensor fields $\mathbf{T}_{1(\mathbf{X})} = \left(\frac{\nabla I}{\|\nabla I\|}\right)\left(\frac{\nabla I}{\|\nabla I\|}\right)^T$ (purely anisotropic) and $\mathbf{T}_{2(x)} = \mathbb{I}_d$ (purely isotropic) lead both to the heat equation $\frac{\partial I}{\partial t} = \Delta I$ which has obviously an isotropic smoothing behavior. Different tensors fields $\mathbf{T}$ with different shapes (isotropic or anisotropic) may define the same regularization behavior. This is due to the fact that the divergence implicitly introduces a dependance on the *spatial variations* of the tensor field $\mathbf{T}$, so it hampers the design of a pointwise smoothing behavior.

Trace-Based PDEs

Alternative PDE-based regularization approaches have been proposed in [3, 51, 68, 69, 78] in order to smooth an image directed by a local smoothing geometry. They are inspirit very similar to the divergence equation (69), but based on a *trace* operator:

$$\forall i = 1, \ldots, n, \quad \frac{\partial I_i}{\partial t} = \text{trace}\,(\mathbf{T}\mathbf{H}_i) \qquad \text{with } \mathbf{H}_i = \begin{pmatrix} \frac{\partial^2 I_i}{\partial x^2} & \frac{\partial^2 I_i}{\partial x \partial y} \\ \frac{\partial^2 I_i}{\partial x \partial y} & \frac{\partial^2 I_i}{\partial y^2} \end{pmatrix} \tag{71}$$

$\mathbf{H}_i$ stands for the Hessian of I_i. Equation (71) is in fact nothing more than a tensor-based expression of the PDE $\frac{\partial \mathbf{I}}{\partial t} = f^{-}_{(\lambda^-,\lambda^+)}\, \mathbf{I}_{\theta^-\theta^-} + f^{+}_{(\lambda^-,\lambda^+)}\, \mathbf{I}_{\theta^+\theta^+}$ where $\mathbf{I}_{\theta^-\theta^-} = \frac{\partial^2 \mathbf{I}}{\partial \theta^{-2}}$. This PDE can be viewed as a simultaneous combination of two orthogonally oriented and weighted 1D Laplacians. In case of multivalued images, each channel I_i of $\mathbf{I}$ is here also coherently smoothed with the same tensor field $\mathbf{T}$. As demonstrated in [78], the evolution of Eq. (71) has a geometric meaning in terms of local linear filtering: it may be seen locally as the application of very small convolutions around each point $\mathbf{X}$ with a Gaussian mask $G_t^{\mathbf{T}}$ *oriented* by the tensor $\mathbf{T}_{(x)}$:

$$G^{\mathbf{T}}_{t(\mathbf{X})} = \frac{1}{4\pi t} \exp\left(-\frac{\mathbf{X}^T\ \mathbf{T}^{-1}\ \mathbf{X}}{4t} \right)$$

This ensures that the smoothing performed by (71) is indeed oriented along the predefined smoothing geometry $\mathbf{T}$. As the trace is not a differential operator, the spatial variation of $\mathbf{T}$ does not trouble the diffusion directions here and two different tensor fields will necessarily lead to different smoothing behaviors. Under certain conditions, the divergence PDE (69) may be also developed as a trace formulation (71). But in this case, the tensors inside the trace and the divergence *are not the same* [78]. Note that trace-based equations (71) are rather directly connected to functional minimizations, especially when considering the multivalued case. For scalar-valued images ($n = 1$), some correspondences are known anyway [3, 19, 51].

Curvature-Preserving PDEs

Basically, the divergence and trace Eqs. (69) and (71) locally behave as oriented Gaussian smoothing whose strengths and orientations are directly related to the tensors $\mathbf{T}_{(x)}$. But on curved structures (like corners), this behavior is not desirable: in case of high variations of the edge orientation θ^-, such a smoothing will tend to *round* corners, even by conducting it only along θ^- (an oriented Gaussian is not curved by itself). To avoid this over-smoothing effect, regularization PDEs may try to stop their action on corners (by vanishing tensors $\mathbf{T}_{(x)}$ there, i.e., $f^- = f^+ = 0$), but this implies the detection of curved structures on images that are themselves noisy or corrupted. This is generally a hard task.

To overcome this problem, curvature-preserving regularization PDEs have been introduced in [77]. We illustrate the general idea of these equations by considering the simplest case of image smoothing along a single direction, i.e., a *vector field* $\mathbf{w} : \Omega \to \mathbb{R}^2$ instead of a tensor-valued one $\mathbf{T}$. The two spatial components of $\mathbf{w}$ are denoted $\mathbf{w}_{(x)} = (u_{(x)} v_{(x)})^T$.

The curvature-preserving regularization PDE that smoothes $\mathbf{I}$ along $\mathbf{w}$ is defined as

$$\forall i = 1, \ldots, n, \quad \frac{\partial I_i}{\partial t} = \operatorname{trace}\left(\mathbf{w}\mathbf{w}^T \mathbf{H}_i\right) + \nabla I_i^T \mathbf{J}_\mathbf{w} \mathbf{w} \qquad \text{with } \mathbf{J}_\mathbf{w} = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{pmatrix} \tag{72}$$

where $\mathbf{J}_\mathbf{w}$ stands for the Jacobian of $\mathbf{w}$. Equation (72) simply adds a term $\nabla I_i^T \mathbf{J}_\mathbf{w} \mathbf{w}$ to the corresponding trace-based PDE (71) that would smooth $\mathbf{I}$ along $\mathbf{w}$. This term naturally depends on the variation of the vector field $\mathbf{w}$. Actually, it has been demonstrated in [77] that Eq. (72) is equivalent to the application of this one-dimensional PDE flow:

$$\frac{\partial I_i(\mathcal{C}_{(a)})}{\partial t} = \frac{\partial^2 I_i(\mathcal{C}_{(a)})}{\partial a^2} \qquad \text{with } \begin{cases} \mathcal{C}^\mathbf{X}_{(0)} & = \quad \mathbf{X} \\ \frac{\partial \mathcal{C}^\mathbf{X}_{(a)}}{\partial a} & = \mathbf{w}\left(\mathcal{C}^\mathbf{X}_{(a)}\right) \end{cases} \tag{73}$$

where $\mathcal{C}^\mathbf{X}_{(a)}$ is the streamline curve of $\mathbf{w}$, starting from $\mathbf{X}$ and parameterized by $a \in \mathbb{R}$. Thus, Eq. (73) is nothing more than the *one-dimensional heat flow constrained on the streamline curve* $\mathcal{C}$. This is indeed very different from a heat flow *oriented* by $\mathbf{w}$, as in the formulation $\frac{\partial I_i}{\partial t} = \frac{\partial^2 I_i}{\partial \mathbf{w}^2}$ since the curvatures of the streamline of $\mathbf{w}$ are now implicitly taken into account. In particular, Eq. (73) has the interesting property to vanish when the image intensities are constant on the streamline $\mathcal{C}^\mathbf{X}$, whatever the curvature of $\mathcal{C}^\mathbf{X}$ is. So, defining a field $\mathbf{w}$ that is tangent everywhere to the image structures allows the preservation of these structures during the regularization process, even if they are curved (such as corners).

Moreover, as Eq. (73) is a 1D heat flow on a streamline $\mathcal{C}^\mathbf{X}$, its solution at time dt can be estimated by convolving the image signal lying on the streamline $\mathcal{C}^\mathbf{X}$ by a 1D Gaussian kernel [50]:

$$\forall \mathbf{X} \in \Omega, \qquad \mathbf{I}^{[dt]}_{(\mathbf{X})} = \int_{-\infty}^{+\infty} \mathbf{I}^{[t=0]}\left(\mathcal{C}^\mathbf{X}_{(p)}\right) \; G^{dt}(p) \; dp \tag{74}$$

This formulation is very close to the line integral convolution (LIC) framework [17], which has been introduced as a visualization technique to render a textured image representing a 2D vector field $\mathbf{w}$. As we are considering diffusion equations here, the weighting function in Eq. (74) is naturally Gaussian. This geometric interpretation particularly allows to implement curvature-preserving PDEs (74) using Runge-

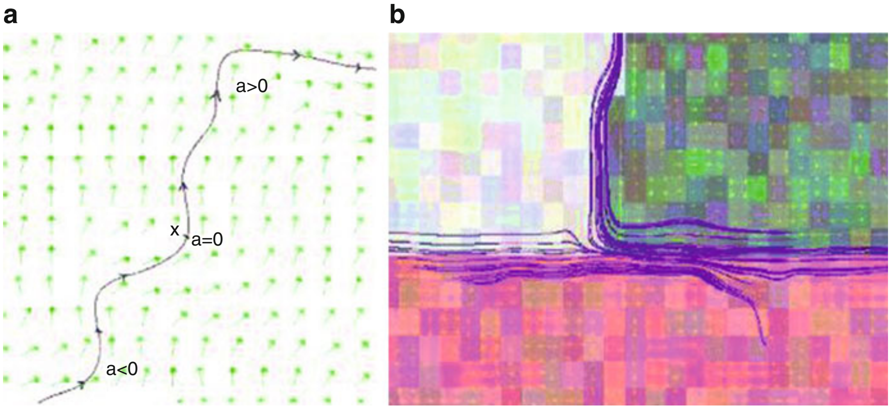


Fig. 27 Streamline $\mathcal{C}^X$ of various vector fields $\mathbf{w}:\Omega \to \mathbb{R}^2$. (**a**) Streamline of a general field **w**. (**b**) Example of streamlines when **w** is the lowest eigenvector of the smoothed structure tensor $\mathbf{G}_\sigma$ (one block is one color pixel)

Kutta estimations of the streamline geometries, leading to sub-pixel precision of the smoothing process.

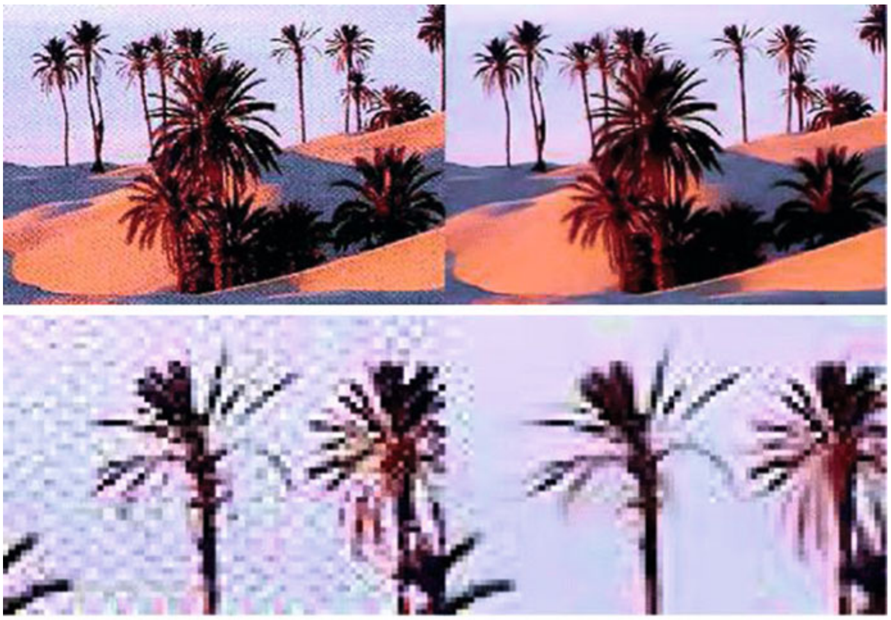
This single-direction smoothing PDE (72) can be easily extended to deal with a tensor-valued geometry $\mathbf{T}$: $\Omega \to$ P(2), in order to be able to represent both *anisotropic* or *isotropic* regularization behaviors. This is done by decomposing the tensor field $\mathbf{T}$ as the sum of several single-directional tensors, i.e., $\mathbf{T} = \frac{2}{\pi}\int_{\alpha=0}^{\pi}\left(\sqrt{\mathbf{T}}a_\alpha\right)\left(\sqrt{\mathbf{T}}a_\alpha\right)^T d\alpha$, where $a_\alpha = \cos\alpha \ \sin\alpha^T$. This naturally suggests to decompose a tensor-driven regularization process into a sum of single-direction smoothing processes, each of them being expressed as a curvature-preserving PDE. As a result, the corresponding curvature-preserving PDE directed by a tensor field $\mathbf{T}$ is

$$\forall i = 1,\ldots,n, \qquad \frac{\partial I_i}{\partial t} = \text{trace}(\mathbf{T}\mathbf{H}_i) + \frac{2}{\pi}\nabla I_i^T \int_{\alpha=0}^{\pi} \mathbf{J}_{\sqrt{\mathbf{T}}a_a}\sqrt{\mathbf{T}}a_a \, d\alpha \tag{75}$$

When $\mathbf{T}$ is locally isotropic (on homogeneous region), Eq. (75) is similar to a 2D heat flow, while when $\mathbf{T}$ is locally anistropic (on an image contour), it behaves as a 1D heat flow on the streamline curve following the contour path, thus taking care of its curvature (Fig. 27).

Applications

Some application results are presented here, mainly based on the use of the curvature-preserving PDEs (75). A specific diffusion tensor field $\mathbf{T}$ has been used to adapt the smoothing behavior to each targeted application.


Noisy color image (*left*), denoised image (*right*) by curvature-preserving PDE (50.75)

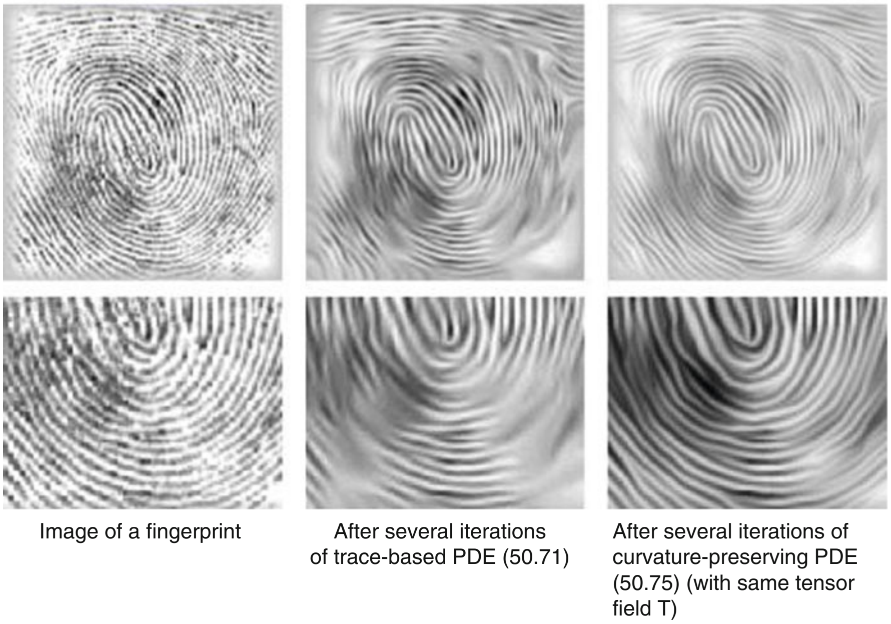
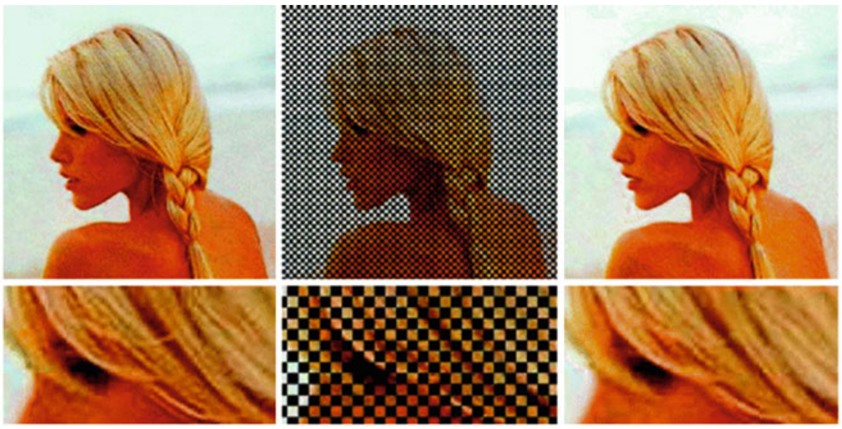
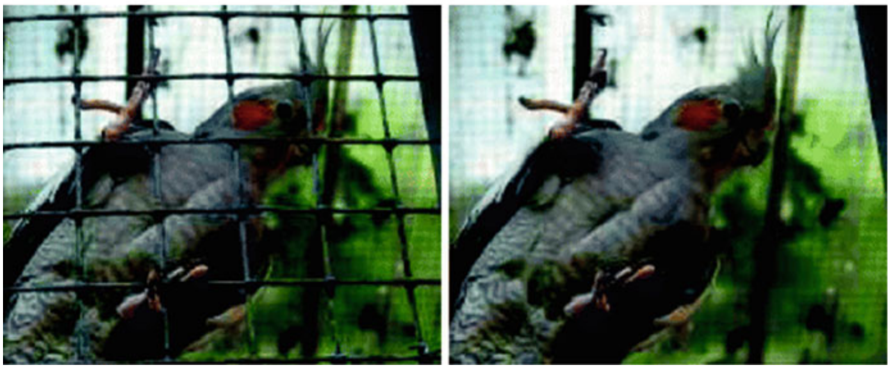


Image of a fingerprint
After several iterations of trace-based PDE (50.71)
After several iterations of curvature-preserving PDE (50.75) (with same tensor field T)

Fig. 28 Using PDE-based smoothing to regularize color and grayscale images



Original color image (*left*), image with 50% pixel removed (*middle*), reconstructed using PDE (50.75) (*right*)



Original color image (*left*), reconstructed using PDE (50.75) (*right*) (the in painting mask covers the cage)

Fig. 29 Image inpainting using PDE-based regularization techniques

Color Image Denoising

Image denoising is a direct application of regularization methods. Sensor inaccuracies, digital quantifications, or compression artifacts are indeed some of the various noise sources that can affect a digital image, and suppressing them is a desirable goal. Figure 28 illustrates how curvature-preserving PDEs (75) can be successfully applied to remove such noise artifacts while preserving the thin structures of the processed images. The tensor field $\mathbf{T}$ is chosen as in Eq. (68).

Color Image Inpainting

Image inpainting consists in filling in missing (user-defined) image regions by guessing pixel values such that the reconstructed image still looks natural. Basically, the user provides one color image $\mathbf{I} : \Omega \to \mathbb{R}^3$ and one *mask* image $M : \Omega \to \{0, 1\}$.

The inpainting algorithm must fill in the regions where $M(X) = 1$, by means of some intelligent interpolations. Image inpainting using diffusion PDEs has been proposed, for instance, in [10, 18, 78]. Inpainting is a direct application of our proposed curvature-preserving PDE (75), where the diffusion equation is applied only on the regions to inpaint, allowing the neighbor pixels to diffuse inside these regions in an anisotropic way (Fig. 29).

Visualization of Vector and Tensor Fields

Regularization PDEs such as (69), (71), and (75) can be also used to visualize a vector field $\mathbf{w} : \Omega \rightarrow \mathbb{R}^2$ or a tensor field $\mathbf{G}$: $\Omega \rightarrow$ P(2); see also Sect. 5. The idea is to smooth an originally pure noisy image using a diffusion tensor field $\mathbf{T}$ which is chosen to be $\mathbf{T} = \mathbf{w}\mathbf{w}^T$ or $\mathbf{T} = \mathbf{G}$ or other variations as long as the smoothing geometry is indeed directed by the field we want to visualize. Whereas the PDE evolution time t goes by, more global structures of the considered fields appear, i.e., a visualization *scale-space* is constructed. The same PDE-based visualization technique allows to display interesting global rendering of DT-MRI volumes (medical imaging) displaying "stuffed" views of the fiber map (Fig. 30).

7 Conclusion

This chapter presented a selection of possible approaches for systematic treatment of multidimensional data sets and algorithms based on differential methods. Such data sets may be the result of some image processing, for instance, a three-dimensional stack of CT slices in medical imaging used to extract a triangular surface representing bones. Visual data analysis is particularly important for big data. Many such data sources originate in computational sciences, produced by algorithms based on differential methods. Utilizing the same concepts for image processing and multidimensional data in general allows generalization of methods and increased software reusability and applicability. Section 2 reviews a mathematically founded model to structure multidimensional data covering a wide category of data types. Since there is no commonly agreed standard in the scientific community on how to lay out multidimensional data for computational purposes, a multitude of alternative models exist. For a specific application, it is subject of investigation whether some specific model would fit all the respective requirements. Section 3 delves deeper into the modeling of mathematical operators using computational data structures. A mathematical algorithm must be cast into a discretized form in order to be applicable in a numerical code. Differential forms are a mathematical abstraction allowing coordinate-free formulations of partial differential equations, which are the basis of many physical and engineering methods, as well as image processing. Geometric Algebra, reviewed in Sect. 4, extends the notions of the commonly known vector algebra to form a full system of algebraic operations to include (among others) the notion of "dividing vectors." While unfamiliar at first, the result is a visually intuitive way to phrase complex algebraic operations, enabling better insight and more efficient implementation as compared to "ad hoc" approaches. An important

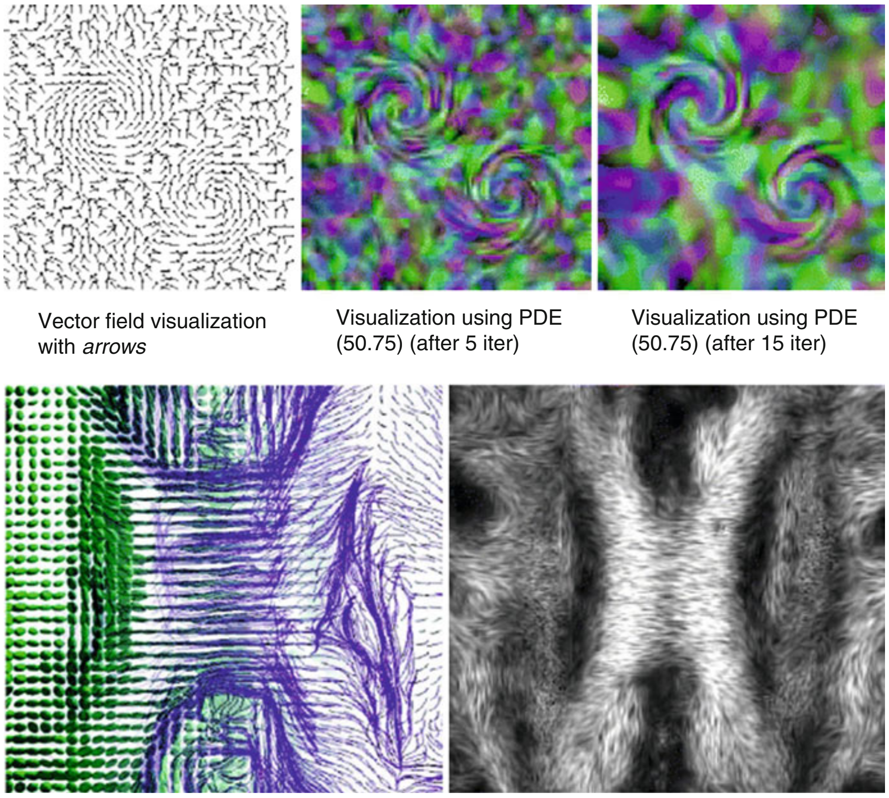

Vector field visualization
with *arrows*
Visualization using PDE
(50.75) (after 5 iter)
Visualization using PDE
(50.75) (after 15 iter)
Tensor field displayed with ellipsoids
(*left*) and tracked fibers (*right*)
Tensor field rendered using a PDE approach
(50.75)

Fig. 30 Visualization of vector and tensor fields using PDEs

application of differential methods for multidimensional data is the investigation of features in vector fields. While primarily of interest to computational sciences such as computational fluid dynamics, identifying topological features in vector fields may well apply to color images with RGB channels and gradients of grayscale images or even more to stacks of images such as three-dimensional data and animation sequences. Section 5 presents some basics for feature detections in such data sets. While the application to animation sequences is beyond the scope of this chapter, utilizing the topological skeleton of some data set may well have potential for data compression or automatizing algorithms for motion pictures. Partial differential equations (PDEs) furthermore allow for direct improvement of image quality and feature reconstruction, as explained in the algorithms presented in Sect. 6. PDEs describing diffusion are particularly suitable for reducing noise in images, recovering lost features, as well as the direct visualization of vector and tensor fields. The set of presented algorithms in this chapter is still subject of active research and neither a comprehensive nor the only reasonable approach; rather, the various aspects covered provide inspirations covering many scientific disciplines under one hood.

Cross-References

▸ Tomography
▸ Mathematical Methods in PET and SPECT Imaging
▸ Mathematics of Electron Tomography

References

1. Abłamowicz, R., Fauser, B.: Clifford/bigebra, a maple package for Clifford (co)algebra computations. Available at http://www.math.tntech.edu/rafal/. © 1996–2009, RA&BF (2009)
2. Alvarez, L., Guichard, F., Lions, P.L., Morel, J.M.: Axioms and fundamental equations of image processing. Arch. Ration. Mech. Anal. **123**(3), 199–257 (1993)
3. Aubert, G., Kornprobst, P.: Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations. Applied Mathematical Sciences, vol. 147. Springer, New York (2002)
4. Barash, D.: A fundamental relationship between bilateral filtering, adaptive smoothing and the nonlinear diffusion equation. IEEE Trans. Pattern Anal. Mach. Intell. **24**(6), 844 (2002)
5. Bayro-Corrochano, E., Vallejo, R., Arana-Daniel, N.: Geometric preprocessing, geometric feedforward neural networks and Clifford support vector machines for visual learning. Spec. Issue J. Neurocomput. **67**, 54–105 (2005)
6. Becker, J., Preusser, T., Rumpf, M.: PDE methods in flow simulation post processing. Comput. Vis. Sci. **3**(3), 159–167 (2000)
7. Benger, W.: Visualization of general relativistic tensor fields via a fiber bundle data model. PhD thesis, FU Berlin (2004)
8. Benger, W.: Colliding galaxies, rotating neutron stars and merging black holes – visualising high dimensional data sets on arbitrary meshes. N. J. Phys. **10** (2008). http://stacks.iop.org/1367-2630/10/125004
9. Benger, W., Ritter, M, Acharya, S., Roy, S., Jijao, F.: Fiberbundle-based visualization of a stir tank fluid. In: WSCG 2009, Plzen (2009)
10. Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C.: Image inpainting. In: ACM SIGGRAPH, International Conference on Graphics and Interactive Techniques, New Orleans, pp. 417–424 (2000)
11. Black, M.J., Sapiro, G., Marimont, D.H., Heeger, D.: Robust anisotropic diffusion. IEEE Trans. Image Process. **7**(3), 421–432 (1998)
12. Bochev, P., Hyman, M.: Principles of compatible discretizations. In: The IMA Hot Topics Workshop on Compatible Discretizations, University of Minnesota, 11–15 May 2004. IMA, vol. 142, pp. 89–120. Springer (2006)
13. Brouwer, L.: Zur Invarianz des n-dimensionalen Gebiets. Math. Ann. **71**, 305–313 (1912)
14. Buchholz, S., Hitzer, E.M.S., Tachibana, K.: Optimal learning rates for Clifford neurons. In: International Conference on Artificial Neural Networks, Porto, vol. 1, pp. 864–873, 9–13 (2007)
15. Butler, D.M., Bryson, S.: Vector bundle classes form a powerful tool for scientific visualization. Comput. Phys. **6**, 576–584 (1992)
16. Butler, D.M., Pendley, M.H.: A visualization model based on the mathematics of fiber bundles. Comput. Phys. **3**(5), 45–51 (1989)

17. Cabral, B., Leedom, L.C.: Imaging vector fields using line integral convolution. In: SIGGRAPH'93, in Computer Graphics, Anaheim, vol. 27, pp. 263–272 (1993)
18. Chan, T., Shen, J.: Non-texture inpaintings by curvature-driven diffusions. J. Vis. Commun. Image Represent. **12**(4), 436–449 (2001)
19. Charbonnier, P., Blanc-Féraud, L., Aubert, G., Barlaud, M.: Deterministic edge-preserving regularization in computed imaging. IEEE Trans. Image Process. **6**(2), 298–311 (1997)
20. Clifford, W.K.: Applications of Grassmann's extensive algebra. In: Tucker, R. (ed.) Mathematical Papers, pp. 266–276. Macmillian, London (1882)
21. Clifford, W.K.: On the classification of geometric algebras. In: Tucker, R. (ed) Mathematical Papers, pp. 397–401. Macmillian, London (1882)
22. Di Zenzo, S.: A note on the gradient of a multi-image. Comput. Vis. Graph. Image Process. **33**, 116–125 (1986)
23. Dorst, L., Fontijne, D., Mann, S.: Geometric Algebra for Computer Science, an Object-Oriented Approach to Geometry. Morgan Kaufman, San Mateo (2007)
24. Ebling, J.: Clifford Fourier transform on vector fields. IEEE Trans. Vis. Comput. Graph. **11**(4), 469–479. IEEE member Scheuermann, Gerik (2005)
25. Firby, P., Gardiner, C.: Surface Topology, chap. 7, pp. 115–135. Ellis Horwood, Chichester (1982). Vector Fields on Surfaces
26. Fontijne, D.: Efficient implementation of geometric algebra. PhD thesis, University of Amsterdam (2007)
27. Fontijne, D., Bouma, T., Dorst, L.: Gaigen: a geometric algebra implementation generator. http://www.science.uva.nl/ga/gaigen (2005)
28. Fontijne, D., Dorst, L.: Modeling 3D Euclidean geometry. IEEE Comput. Graph. Appl. **23**(2), 68–78 (2003)
29. Garth, C., Tricoche, X., Scheuermann, G.: Tracking of vector field singularities in unstructured 3D time-dependent datasets. In: Proceedings of the IEEE Visualization, Austin, pp. 329–336 (2004)
30. Globus, A., Levit, C., Lasinski, T.: A tool for visualizing the topology of threedimensional vector fields. In: Proceedings of the IEEE Visualization '91, San Diego, pp. 33–40 (1991)
31. Gottlieb, D.H.: Vector fields and classical theorems of topology. Rend. Semin. Mat. Fisico, Milano **60**, 193–203 (1990)
32. Gottlieb, D.H.: All the way with Gauss-Bonnet and the sociology of mathematics. Am. Math. Mon. **103**(6), 457–469 (1996)
33. Gross, P., Kotiuga, P.R.: Electromagnetic Theory and Computation: A Topological Approach. Cambridge University Press, Cambridge (2004)
34. Hart, J.: Using the CW-complex to represent the topological structure of implicit surfaces and solids. In: Implicit Surfaces '99, Eurographics/SIGGRAPH, Bordeaux, 13–15 Sept 1999, pp. 107–112
35. Hatcher, A.: Algebraic Topology. Cambridge University Press, Cambridge (2002)
36. Hauser, H., Gröller, E.: Thorough insights by enhanced visualization of flow topology. In: 9th International Symposium on Flow Visualization, Edinburgh (2000). http://www.cg.tuwien.ac.at/research/publications/2000/Hauser-2000-Tho/
37. Helman, J., Hesselink, L.: Representation and display of vector field topology in fluid flow data sets. IEEE Comput. **22**(8), 27–36 (1989)
38. Helman, J., Hesselink, L.: Visualizing vector field topology in fluid flows. IEEE Comput. Graph. Appl. **11**, 36–46 (1991)
39. Hestenes, D.: New Foundations for Classical Mechanics. Reidel, Dordrecht (1986)
40. Hestenes, D., Sobczyk, G.: Clifford Algebra to Geometric Calculus: A Unified Language for Mathematics and Physics. Reidel, Dordrecht (1984)
41. Hildenbrand, D., Fontijne, D., Perwass, C., Dorst, L.: Tutorial geometric algebra and its application to computer graphics. In: Eurographics Conference, Grenoble (2004)
42. Hildenbrand, D., Fontijne, D., Wang, Y., Alexa, M., Dorst, L.: Competitive runtime performance for inverse kinematics algorithms using conformal geometric algebra. In: Eurographics Conference, Vienna (2006)

43. Hildenbrand, D., Lange, H., Stock, F., Koch, A.: Efficient inverse kinematics algorithm based on conformal geometric algebra using reconfigurable hardware. In: GRAPP Conference, Madeira (2008)
44. Hildenbrand, D., Pitt, J.: The Gaalop home page. http://www.gaalop.de (2008)
45. Hocking, J., Young, G.: Topology. Addison-Wesley/Dover, New York (1961)
46. Hunt, J.: Vorticity and vortex dynamics in complex turbulent flows. Proc. CANCAM Trans. Can. Soc. Mech. Eng. **11**, 21 (1987)
47. Jeong, J., Hussain, F.: On the identification of a vortex. J. Fluid Mech. **285**, 69–94 (1995)
48. Kenwright, D., Henze, C., Levit, C.: Feature extraction of separation and attachment lines. IEEE Trans. Vis. Comput. Graph. **5**(2), 135–144 (1999)
49. Kimmel, R., Malladi, R., Sochen, N.: Images as embedded maps and minimal surfaces: movies, color, texture, and volumetric medical images. Int. J. Comput. Vis. **39**(2), 111–129 (2000). doi:10.1023/A:1008171026419
50. Koenderink, J.J.: The structure of images. Biol. Cybern. **50**, 363–370 (1984)
51. Kornprobst, P., Deriche, R., Aubert, G.: Non-linear operators in image restoration. In: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97), San Juan, 17–19 June 1997, p. 325. IEEE Computer Society, Washington, DC (1997)
52. Lindeberg, T.: Scale-Space Theory in Computer Vision. Kluwer Academic, Dordrecht (1994)
53. Löffelmann, H., Doleisch, H., Gröller, E.: Visualizing dynamical systems near critical points. In: Spring Conference on Computer Graphics and Its Applications, Budmerice, pp. 175–184 (1998)
54. Mann, S., Rockwood, A.: Computing singularities of 3D vector fields with geometric algebra. In: Proceedings of the IEEE Visualization, Boston, pp. 283–289 (2002)
55. Mattiussi, C.: The geometry of time-stepping. In: Teixeira, F.L. (ed.) Geometric Methods in Computational Electromagnetics. PIER, vol. 32, pp. 123–149. EMW, Cambridge (2001)
56. McCormick, B., DeFanti, T., Brown, M.: Visualization in scientific computing-a synopsis. IEEE Comput. Graph. Appl. **7**(7), 61–70 (1987). doi:10.1109/MCG.1987.277014
57. Naeve, A., Rockwood, A.: Course 53 geometric algebra. In: SIGGRAPH Conference, Los Angeles (2001)
58. Nielsen, M., Florack, L., Deriche, R.: Regularization, scale-space and edge detection filters. J. Math. Imaging Vis. **7**(4), 291–308 (1997)
59. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. IEEE Trans. Pattern Anal. Mach. Intell. **12**(7), 629–639 (1990)
60. Perwass, C.: The CLU home page. http://www.clucalc.info (2005)
61. Perwass, C.: Geometric Algebra with Applications in Engineering. Springer, Berlin (2009)
62. Petsche, H.-J.: The Grassmann bicentennial conference home page. /http://www.uni-potsdam.de/,u/philosophie/grassmann/Papers.htm (2009)
63. Pham, M.T., Tachibana, K., Hitzer, E.M.S., Yoshikawa, T., Furuhashi, T.: Classification and clustering of spatial patterns with geometric algebra. In: AGACSE Conference, Leipzig (2008)
64. Preußer, T., Rumpf, M.: Anisotropic nonlinear diffusion in flow visualization. In: Proceedings of the Conference on Visualization '99: Celebrating Ten Years. IEEE Visualization, San Francisco, pp. 325–332. IEEE Computer Society, Los Alamitos (1999)
65. Reyes-Lozano, L., Medioni, G., Bayro-Corrochano, E.: Registration of 3d points using geometric algebra and tensor voting. J. Comput. Vis. **75**(3), 351–369 (2007)
66. Rosenhahn, B., Sommer, G.: Pose estimation in conformal geometric algebra. J. Math. Imaging Vis. **22**, 27–70 (2005)
67. Rudin, L., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Physica D **60**, 259–268 (1992)
68. Sapiro, G.: Geometric Partial Differential Equations and Image Analysis. Cambridge University Press, Cambridge (2001)
69. Sapiro, G., Ringach, D.L.: Anisotropic diffusion of multi-valued images with applications to color filtering. IEEE Trans. Image Process. **5**(11), 1582–1585 (1996)
70. Stalling, D., Steinke, T.: Visualization of vector fields in quantum chemistry. Technical report, ZIB m-96–01 (1996)

71. The homepage of geomerics ltd. http://www.geomerics.com. Last visited 2015
72. Theisel, H., Weinkauf, T., Hege, H.-C., Seidel, H.-P.: Saddle connectors – an approach to visualizing the topological skeleton of complex 3D vector fields. In: Proceedings of the IEEE Visualization, Seattle, pp. 225–232 (2003)
73. Theisel, H., Weinkauf, T., Hege, H.-C., Seidel, H.-P.: Grid-independent detection of closed stream lines in 2D vector fields. In: Proceedings of the Vision, Modeling and Visualization 2004, Standford, 16–18 Nov 2004, pp. 421–428. http://www.courant.nyu.edu/~weinkauf/publications/bibtex/theise104b.bib
74. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: Proceedings of the Sixth International Conference on Computer Vision (ICCV), Bombay, 04–07 Jan 1998, p. 839. IEEE Computer Society, Washington, DC (1998)
75. Tonti, E.: The reason for analogies between physical theories. Appl. Math. Model. **1**(1), 37–50 (1976/1977)
76. Treinish, L.A.: Data explorer data model. http://www.research.ibm.com/people/1/lloydt/dm/dx/dx_dm.htm (1997)
77. Tschumperlé, D.: Fast anisotropic smoothing of multi-valued images using curvature- reserving PDE's. Int. J. Comput. Vis. **68**(1), 65–82 (2006). ISSN:0920–5691
78. Tschumperlé, D., Deriche, R.: Vector-valued image regularization with PDE's: a common framework for different applications. IEEE Trans. Pattern Anal. Mach. Intell. **27**(4), 506–517 (2005)
79. Veldhuizen, T.: Using C++ template metaprograms. C++ Rep. **7**(4), 36–43 (1995). Reprinted in C++ Gems, ed. Stanley Lippman
80. Vemuri, B.C., Chen, Y., Rao, M., McGraw, T., Wang, Z., Mareci, T.: Fiber tract mapping from diffusion tensor MRI. In: Proceedings of the IEEE Workshop on Variational and Level Set Methods (VLSM'01), Vancouver, 13 July 2001, p. 81. IEEE Computer Society, Washington, DC (2001)
81. Venkataraman, S., Benger, W., Long, A., Byungil Jeong, L.R.: Visualizing Hurricane Katrina – large data management, rendering and display challenges. In: GRAPHITE 2006, Kuala Lumpur, 29 Nov–2 Dec 2006
82. Weickert, J.: Anisotropic Diffusion in Image Processing. Teubner-Verlag, Stuttgart (1998)
83. Weickert, J.: Coherence-enhancing diffusion of colour images. Image Vis. Comput. **17**, 199–210 (1999)
84. Weinkauf, T.: Extraction of topological structures in 2D and 3D vector fields. PhD thesis, University Magdeburg. http://tinoweinkauf.net/ (2008)
85. Weinkauf, T., Theisel, H., Hege, H.-C., Seidel, H.-P.: Boundary switch connectors for topological visualization of complex 3D vector fields. In: Data Visualization 2004. Proceedings of the VisSym 2004, Konstanz, 19–21 May 2004, pp. 183–192. http://www.courant.nyu.edu/~weinkauf/publications/bibtex/weinkauf04a.bib
86. Zomorodian, A.J.: Topology for Computing. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge/New York (2005). First published 2005, reprinted 2009