Flow Field Clustering via Algebraic Multigrid

M. Griebel* T. Preusser† M. Rumpf‡ M. A. Schweitzer* A. Telea§

ABSTRACT

We present a novel multiscale approach for flow visualization. We define a local alignment tensor that encodes a measure for alignment to the direction of a given flow field. This tensor induces an anisotropic differential operator on the flow domain, which is discretized with a standard finite element technique. The entries of the corresponding stiffness matrix represent the anisotropically weighted couplings of adjacent nodes of the domain mesh. We use an algebraic multigrid algorithm to generate a hierarchy of fine to coarse descriptions for the above coupling data. This hierarchy comprises a set of coarse grid nodes, a multiscale of basis functions and their corresponding supports. We use these supports to obtain a multilevel decomposition of the flow structure. Standard streamline icons are used to visualize this decomposition at any user-selected level of detail. The method provides a single framework for vector field decomposition independent on the domain dimension or mesh type. Applications are shown in 2D, for flow fields on curved surfaces, and for 3D volumetric flow fields.

Keywords: algebraic multigrid, multiscale visualization, flow visualization

1 INTRODUCTION

Present computing hardware and efficient numerical algorithms enable large scientific simulations which deliver enormous data amounts. Various visualization strategies have been proposed to represent such data in an intuitively understandable way. The larger and more complex the simulation results become, the stronger is the need for a suitable multiscale visualization technique. Simplified data representations, useful to show global patterns, can be gradually refined for further insight. Moreover, different viewers need different representations. Numerical experts might want to see the full detail data. Technological experts might want to see certain features such as vortices. Other users may desire a simplified presentation.

Clustering, well-known from statistics, provides such a multiscale approach. Data are grouped in successively larger sets which have a strong internal correlation. Many techniques are available for scattered and scalar data, e.g. based on wavelet or Fourier analysis. However, for vector data, only few multiscale visualization methods are available. Regular subsampling, still commonly used, is well-known to produce alias effects. A second class of methods represents the vector field by equally spaced streamline icons, with user-prescribed spacing. Different heuristics [16] and energy minimization stragegies [26] are used to compute the icons' positions.

Several multiscale clustering approaches for vector data have been proposed. These methods build a hierarchical clustering tree and visualize the clusters with curved arrow icons. Heckel et al. [14] place all data points in a single cluster, which is recursively split in a top-down manner. At each step, the cluster with the strongest discrepancy between streamlines generated by the original field and its approximation by the cluster is bisected by a plane, using principal component analysis. Although this guarantees convex clusters, an accurate representation may require a large cluster count. Telea and Van Wijk [21] place each data point in a cluster. Next, similar clusters are merged bottom-up, using a metric of the difference in position and orientation of the vectors that represent the clusters. The cluster shapes are indirectly constrained by adapting the weights of the metric terms. However, this method may be sensitive to the weight tuning, leading to clusters that are not aligned with the flow.

A separate class of vector field simplification methods explicitly detects features such as saddles, sources, sinks, and vortices. A multiscale simplification is then obtained by successively removing pairs of critical points based on some relevance metric [24]. In [10], a pattern matching approach using Clifford algebra convolution operations is proposed. Vortex features are found by matching a particular vortex mask against the flow field. Similar approaches are presented by [22] and [1]. Here, the multiscale is given in a scale-space fashion by the convolution operator's size. Overall, such methods are quite sensitive to the feature definition ('what is a vortex?') and the detection process used, especially for 3D fields.

A different approach, proposed by [17], decomposes the field into a divergence free, a rotation free, and a harmonic component, which are separately simplified by detecting critical points corresponding to the components' extrema. This method avoids some of the critical point detection problems in standard topological methods. A similar method that also incorporates multiple scales via component filtering has been recently presented by [23].

Finally, we mention the class of physics-based, continuous clustering approaches. These methods simulate a physical process, such as anisotropic diffusion [9, 11], applied to an initial fine-grained, noise-like signal. The initial clusters, represented implicitly by the fine-grained noise, are coarsened and aligned to the flow, which determines the anisotropy of the operator. The diffusion time serves as a multiscale parameter.

We employ a similar idea: We construct a multiscale decomposition via an anisotropic diffusion operator. However, we construct the cluster hierarchy in a strictly algebraic fashion. Given a flow domain, we define a particular anisotropic diffusion tensor based on the flow direction. Discretizing the corresponding anisotropic differential operator using finite elements defines strong (flow-aligned) and weak (flow-orthogonal) couplings between mesh neighbor points. We obtain a finite element stiffness matrix encoding the flow on the finest level. Next, we use the algebraic multigrid (AMG) method, a well-known technique for accelerating the solution of systems of linear equations. AMG computes a sequence of progressively simplified stiffness matrices which approximate the operator as well as possible with a given number of degrees of freedom. Together with each simplified matrix, AMG delivers a set of basis functions associated with the actual degrees of freedom. AMG's coarsening procedure ensures that the basis

*Institute for Numerical Simulation, University of Bonn, Wegelerstr. 6, 53115 Bonn, [griebel,schweitzer]@ins.uni-bonn.de

†Center for Complex Systems and Visualization, University of Bremen, Universitätsallee 29, 28359 Bremen, tp@cevis.uni-bremen.de

‡Institute for Numerical Analysis and Scientific Computing, Duisburg-Essen University, Lotharstrasse 65, 47048 Duisburg, rumpf@math.uni-duisburg.de

§Department of Mathematics and Computer Science, Eindhoven University, Den Dolech 2, Eindhoven, Netherlands, alext@win.tue.nl

functions reflect the operator-encoded coupling and that their supports are flow-aligned. Finer level basis functions reflect finer flow details. Coarser levels contain fewer basis functions, which reflect coarse-scale flow patterns. At any simplification level, we define the domain of a basis function as the subset of its support where this function is maximal among all other basis functions on that level. This yields a flow domain decomposition in flow-aligned disjoint regions, at the desired level of detail. Finally, we visualize these domains either by direct color coding or by using curved arrow icons, as in [21, 11]. Our method shares its conceptual origin with an AMG-based multiscale, feature-sensitive editing tool for surfaces [8].

In summary, our method solves the question "show a vector field with n flow icons, for a given n" not only very effectively but also very efficiently: Given AMG's high computational performance, we can treat 2D and 3D vector fields in a few seconds up to tens of seconds, respectively.

This paper is structured as follows. We first define the operator describing the neighbor mesh point coupling (Sec. 2). Next, we briefly review the algebraic multigrid as a general tool for coarsening discrete differential operators (Sec. 3). Section 4 explains the discretization of our anisotropic diffusion operator. The flow domain decomposition based on the sequence of AMG basis functions is discussed in detail in Sec. 5. Section 6 presents various applications. Section 7 discusses how to select a local level of detail. Finally, we draw conclusions in Section 9.

2 An operator encoding the flow alignment

Any clustering algorithm relies on a given coupling of the objects to be clustered [15]. Our goal is to cluster flow fields, so we first define a local coupling on the flow domain. We desire a strong coupling in the direction of the flow and a weak coupling orthogonal to the flow. Let us consider a stationary flow field

$$v : \Omega \to \mathbb{R}^d$$

on a flow domain $\Omega \subset \mathbb{R}^d$. We first restrict ourselves to the case of a two-dimensional flat domain ($d = 2$) or a three-dimensional volume ($d = 3$). We will generalize our approach to flow fields on curved surfaces (Sec. 6.3).

We encode our coupling in a *local coupling tensor*. Let us suppose that $v \neq 0$ on Ω. Then, at each point $x \in \Omega$, there exists a rotation $B(v(x))$ generating a coordinate transform with $B(v)v = \|v\| e_0$, where $\{e_i\}_{i=1,\cdots,d}$ is the standard basis in $\mathbb{R}^d$. Taking into account this coordinate transform, we define a local coupling tensor

$$a(v) = B(v)^T \begin{pmatrix} \alpha(\|v\|) & \\ & \varepsilon \mathrm{Id}_{d-1} \end{pmatrix} B(v)$$

where $\alpha : \mathbb{R}^+ \to \mathbb{R}^+$ represents the strong coupling in the flow direction, and Id_{d-1} is the identity matrix in dimension $d-1$. In any direction orthogonal to v we have a coupling with weight ε, which is supposed to be small. A good choice for the function $\alpha(\cdot)$ is a monotone function α with $\alpha(0) = \varepsilon$ and $\lim_{s\to\infty} \alpha(s) = \bar{\alpha}$ for some constant $\bar{\alpha} > 1$. This approach is useful even when the magnitude of the vector field spatially varies over several orders. For fields having less variation in their vector magnitudes, and for normalized fields, we use the simple form $\alpha(s) = s + \varepsilon$, with $\varepsilon = 0.001$. A similar coupling tensor was used for flow visualization in [9], where the tensor induced a diffusive growth of flow aligned patterns.

We now define a differential operator $\mathscr{A}$ on the domain Ω, which encodes an anisotropic stiffness described by the pointwise defined tensor a on Ω. For any function $u \in C^2(\Omega)$ we set

$$\mathscr{A} u := -\mathrm{div}(a(v)\nabla u)\,.$$

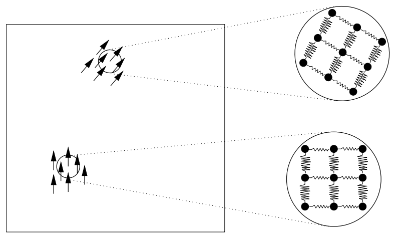
Figure 1: Spring model of local coupling tensor. The thick springs in the two magnified areas indicate larger stiffness, hence strong coupling, as compared to the thinner (weaker) springs.

In other words, the flow domain is considered to be locally stiff in the flow direction and less stiff in the perpendicular direction. This can be depicted using a simple spring model (cf. Fig. 1). Springs are stiff if they couple nodes along a streamline and weak if the spring is perpendicular to the flow. Alternatively, we can describe the role of $\mathscr{A}$ in terms of diffusion. To this end, we consider an anisotropic diffusion process controlled by an operator $\mathscr{A}$ with strong diffusion in direction of the flow field.

We now can ask for a multiscale (fine to coarse) representation of the operator $\mathscr{A}$. Fortunatly, there is a well-developed mathematical theory for such a multiscale representation. In the following, we will discretize $\mathscr{A}$ via piecewise linear finite elements on a triangular (in 2D) or tetrahedral (in 3D) mesh covering the flow domain and then apply a suitable algebraic multigrid method to compute successively coarser representations of the discrete fine grid operator. Finally, we will derive a multiscale domain decomposition from this multiscale of discrete operators. This domain decomposition is our final clustering result. To motivate the later discussion, we next review the basics of algebraic multigrid.

3 A brief review of AMG

Here we give an overview of the basic aspects of the algebraic multigrid algorithm (AMG) and the heuristics which led to its development. We refer to [25] for a detailed introduction. Algebraic multigrid methods were first introduced in the early 1980's [2, 3, 4, 5, 18] for the solution of discrete linear systems $AU = F$ of equations coming from the discretization of a linear differential equation $\mathscr{A} u = f$ on a domain Ω with suitable boundary conditions. Here U is supposed to be a finite element approximation of the continuous solution u and A the finite element stiffness matrix corresponding to $\mathscr{A}$. Finally, F is the corresponding discrete right hand side. The development of AMG was led by the idea to mimick classical (geometric) multigrid methods in applications where a hierarchy of nested meshes is either not available at all, or cannot reflect particular properties such as strength of diffusion of the discretized operator appropriately on coarse grid levels. Hence, one has to work with the matrix A and its algebraic structure. AMG tries to coarsen this matrix independently from any underlying fine grid discretization, yet maintaining optimal computational complexity $O(n)$, where n is the number of rows of the initial matrix. AMG computes a sequence of prolongation matrices P^l which encodes how coarse scale (l) basis functions are combined using the basis functions on the finer scale ($l-1$). This induces a sequence of corresponding matrices A^l, defined by the so-called Galerkin projection $A^l := R^l A^{l-1} P^l$, where the restriction R^l is given as the transpose of the prolongation P^l ($R^l := (P^l)^T$). The prolongation matrices $\{P^l\}_{l=1,\cdots,L}$ are computed using information from the matrix A^{l-1} on the previous level $l-1$ only. The sequence of prolongation matrix allows for the construction of a problem-dependent

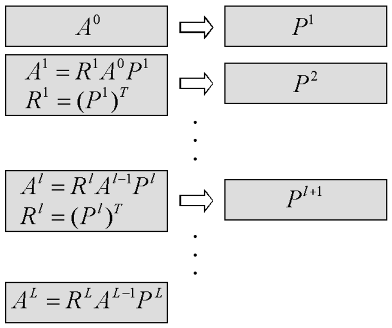

Figure 2: General AMG construction. From the fine scale matrix A^0 input, AMG computes prolongations P^l, restrictions R^l, and coarse scale matrices A^l on successively coarser scales $l = 1, \cdots, L$.

basis $\{\Psi^{l,i}\}$. One constructs a coarser basis $\{\Psi^{l,i}\}$ which captures the appropriate features relevant for the approximation of the corresponding continuous problem, i.e. the underlying differential operator (Sec. 4). The theory and design of efficient AMG tools is rather involved. However, our flow clustering requires just basic AMG capabilities. We perform no specific tuning of the AMG for flow clustering. One could use several AMG packages available on the web (see *www.mgnet.org*) in a black-box manner: feed the fine grid matrix $\mathscr{A}$ to the AMG tool and read the produced matrices P^l and A^l.

The general AMG procedure is sketched in Fig. 2. The fundamental ingredient in this AMG construction is the notion of *algebraic smoothness*. Using such a smoothness measure, we can set up a reduced matrix graph from which we can then "merge" fine level basis functions $\Psi^{l-1,i}$ on level $l-1$ appropriately to define the coarse basis $\{\Psi^{l,i}\}$ on level l. This merging is actually encoded in the columns of the prolongation matrix P^l. Common to nowadays algebraic smoothness criteria [18, 6, 7] is the general observation that a simple relaxation scheme – most often Gauss–Seidel smoothing – efficiently damps components in the direction of eigenvectors associated with large eigenvalues. Consequently, the coarse basis functions are chosen such that they deal with the remaining components of an eigenvector decomposition. Given the anisotropic diffusion tensor a of Sec. 4, AMG joins flow-aligned basis functions automatically in the construction of coarser basis functions. For further details on the specific AMG implementation we used, we refer to [12, 13].

4 A MATRIX ENCODING THE FLOW STRUCTURE

Based on our local tensor $a(v)$, which encodes the flow-aligned coupling, we have defined the differential operator $\mathscr{A}$ which globally represents the flow structure in terms of the local couplings on Ω. We consider now a discretization which follows the general finite element paradigm. Let $\mathscr{V}_h$ be the space of piecewise linear finite elements corresponding to a triangulation of Ω with nodes $(X_i)_{i=1,\cdots,n}$. The piecewise linear basis $\{\Phi_i\}_{i=1,\cdots n}$ of hat functions is uniquely defined by the property $\Phi_i(X_j) = \delta_{ij}$ for all vertices X_j, where δ_{ij} is the Kronecker symbol. In the following, we use capital letters for discrete objects to distinguish them from continuous objects denoted with lower case letters. We compute the $n \times n$ finite element stiffness matrix $A = (A_{ij})_{ij}$ where

$$A_{ij} := \int_{\Omega} a(v)\, \nabla\Phi^i \cdot \nabla\Phi^j \, dx.$$

This matrix describes the weighted coupling of adjacent triangulation nodes on the discrete domain using the local coupling tensor a. This matrix is obviously sparse. We expect a non-vanishing entry A_{ij} if and only if the node i has a mesh edge to the node j.

The assembly of A uses the standard finite element procedure. In case of a 2D flow domain defined on a triangle mesh, we start by initializing $A = 0$, followed by a traversal of all triangles T. On each triangle T with nodes P^0, P^1, P^2, we compute a local matrix $(A^{loc}_{ij}(T))_{ij}$, corresponding to all pairings of local nodal basis functions. This local matrix is next added to the corresponding locations in the global matrix A, i. e. for every pair i, j we update $A_{\beta(i),\beta(j)} = A_{\beta(i),\beta(j)} + A^{loc}_{ij}(T)$. Here $\beta(i)$ is defined as the global index of the node with local index i. For the local matrix, we define the local coupling tensor a for every triangle T as $a = a(v(C_T))$, where C_T denotes the barycenter of the triangle T. We obtain for the local matrix:

$$a_{ij}(T) = a(v(C_T))\nabla_T\Phi_i \cdot \nabla_T\Phi_j\,|T|$$

where $|T|$ is the area of T. Given the sparsity of A, we use a compressed row matrix storage, i.e., we store only the nonzero entries and their column indices for every matrix row. 3D tetrahedral meshes are treated in a completely analogous fashion. Similar assembly procedures can be devised for 2D quad meshes and 3D hexahedral meshes, or even mixed element meshes, if desired. Table 1 shows the assembly times for various meshes.

5 MULTISCALE DECOMPOSITION

The matrix A defined in Sec. 4 can be seen as a description of the structure of a flow field v. In particular, flow alignment is encoded in this matrix. The matrix simultaneously represents dominant flow patterns as well as successively finer, more detailed flow structures. At this point, we require a tool able to represent flow patterns in a hierarchical multiscale fashion. We compute this multiscale using the AMG method of Section 3. AMG delivers a set of descriptions of the flow-induced coupling in terms of matrices A^l for $l = 0, \cdots L$, ranging from detailed ($A^0 = A$) to very coarse (A^L). Let us now illustrate how AMG works using two simple examples. Consider the flow fields $v_1(x) = (-1, 1)$ and $v_2(x) = (1, 1)$ on the square domain $\Omega = [-1; 1]^2 \subset \mathbb{R}^2$. This implies a coupling tensors

$$a = B^T \begin{pmatrix} \sqrt{2} + 0.001 & 0 \\ 0 & 0.001 \end{pmatrix} B,$$

where B is a rotation of ∓ 45 degrees, respectively. We then consider the corresponding differential operator $\mathscr{A} = \mathrm{div}(a\nabla\cdot)$ and apply the AMG method to the matrix which results from the discretization of $\mathscr{A}$ on a regular triangulation.

Figure 3 shows the coupling strengths encoded in the matrices A^l for the first three finest levels $l = 0, 1, 2$, for the fields $v_1 = (1, 1)$ and $v_2 = (-1, 1)$, using a blue-to-red colormap. Edges represent (nonzero) matrix entries. For the same fields, Fig. 4 shows selected basis functions on the four coarsest decomposition levels. The sparsity of the finest level (Fig. 3 a,d) corresponds to the regular triangular mesh connectivity. The coarse matrices (Fig. 3 b,c,e,f) are sparse, too. In general, however, their sparsity pattern does not correspond to edges of a coarsened finite element grid. Indeed, the lines in Figs. 3 b,c,e,f, which indicate coarse matrix entries, do not correspond to edges in the initial grid. Furthermore, we see that the coupling strengths differ for the two fields v_1 and v_2, (Fig. 3 a-c vs Fig. 3 d-f). This is due to the relative orientation of the field to the mesh. The field v_1 is perfectly aligned with the mesh structure, whereas there are no mesh edges in the direction of the field v_2. Still, we see in both cases that the strong couplings (red lines) match the field direction. Moreover, the basis function shapes nicely follow the field direction in both cases (Fig. 4). This can also be seen

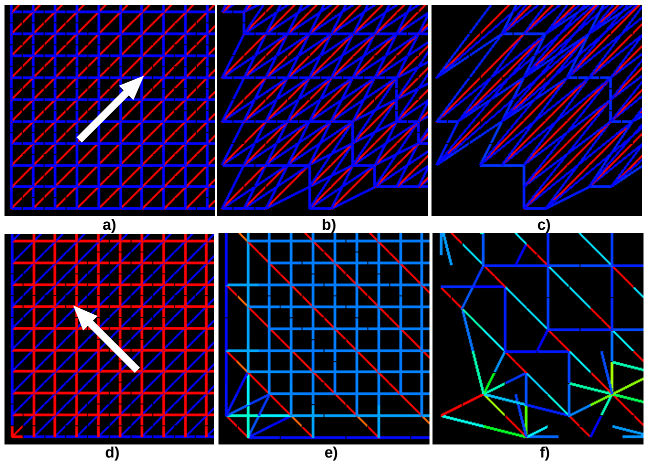

Figure 3: Color-coded coupling strength (zoomed in). Three finest levels (left to right) are shown for the fields $v_1 = (-1,1)$ (bottom row) and $v_2 = (1,1)$ (top row). The white arrows show the field direction.

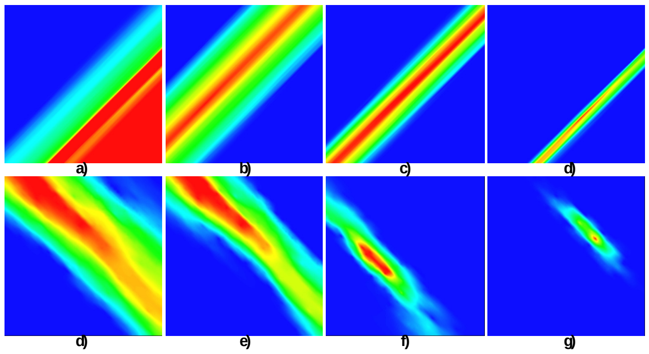

Figure 4: Selected basis functions, on the four coarsest levels, for the fields $v_1 = (-1,1)$ (bottom row) and $v_2 = (1,1)$ (top row)

in Fig. 12 c which shows several basis functions for the more complex flow from Fig. 7.

Let us now explicitly derive the multiscale of basis functions from the sequence of prolongation matrices. When we apply the AMG algorithm (Sec. 3) to the matrix $A \in \mathbb{R}^{n,n}$ introduced in the previous section, we obtain a sequence of prolongation matrices

$$P^l \in \mathbb{R}^{n_{l-1},n_l}.$$

as output, where n_l for $l = 0,\cdots,L$ are the decreasing numbers of remaining unknowns and $n_0 = n$. The entries in each column $i = 1,\cdots,n_l$ of P^l give the coefficients of the linear combination of the finer basis functions $\Psi^{l-1,j}$ for $j = 1,\cdots,n_{l-1}$ corresponsing to the coarser basis function $\Psi^{l,i}$ on level l. In other words, each matrix A^l delivered by the AMG, starting with the initial, finest one $A^0 = A$ down to the coarsest one A^L, approximates the fine grid operator using the (matrix-dependent) basis $\{\Psi^{l,i}\}_{i=1,\cdots,n_l}$:

$$A^l_{ij} = A\overline{\Psi^{l,i}}\cdot\overline{\Psi^{l,j}} = \int_\Omega a(v)\nabla\Psi^{l,i}\cdot\nabla\Psi^{l,j},$$

where $\overline{\Psi^{l,i}}$ is the nodal vector corresponding to the function $\Psi^{l,i}$, i. e. $\Psi^{l,i} = \sum_{j=1,\cdots,n}(\overline{\Psi^{l,i}})_j\Phi^j$ where Φ^j denotes the initial basis functions.

Hence, the following simple recursive recipe can be used to calculate the multiscale of basis functions $\Psi^{l,i}$ (cf. Fig. 4):

$$\Psi^{l,i} := \sum_{j=1,\cdots,n_{l-1}} P^l_{ji}\Psi^{l-1,j} \quad \forall i = 1,\cdots n;\, l = 1,\cdots L$$
$$\Psi^{0,i} := \Phi^i \quad \forall i_1,\cdots n$$

The shapes of the basis functions clearly shows the strength of the local coupling (compare Fig. 4 with Fig. 3). The AMG method clusters vertices along a streamline already on a fine scale, since they are strongly coupled. Vertices not aligned to the flow are clustered on coarser scales, since their coupling is relatively weaker.

As usual with finite elements, the supports of basis functions on a given scale are overlapping. Hence, we need to derive a multiscale of domain decompositions from the set of basis functions to partition the domain into disjoint clusters. Such a domain decomposition

$$\mathscr{D}^l := \{\mathscr{D}^{l,i}\}_{i=1,\cdots,n_l}$$

can easily be defined for every $l = 0,\cdots L$ as follows (cf. Fig.5):

$$\mathscr{D}^{l,i} := \{x \in \Omega \,|\, \Psi^{l,i}(x) \geq \Psi^{l,j}(x)\,\forall j = 1,\cdots,n_l\}$$

In other words, a domain $\mathscr{D}^{l,i}$ on level l is the set of points where the basis $\Psi^{l,i}$ is maximal on that level. Figure 5 shows the domains,

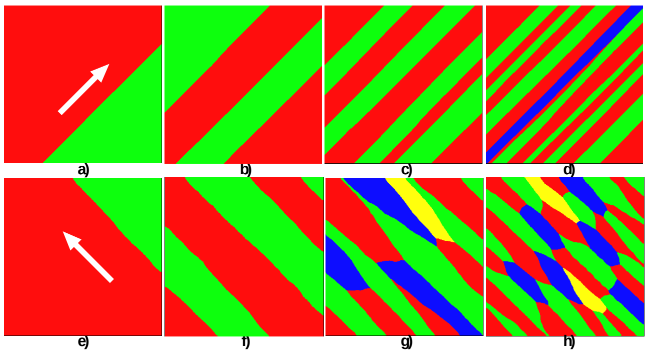

Figure 5: Domain decomposition for the two fields in Figs. 4 and 3

on the four coarsest levels from left to right, for the two sample fields v_1 and v_2 discussed above. We visualize the domains by assigning different colors, picked from a small prescribed color set, to neighboring domains.

Several observations can be made. First, the domains on different scales need not be *strictly* spatially nested – the supports of the shape functions are, but the decomposition arising from the maximal property is not. However, the domains are clearly aligned to the flow field. All domains on a given level l have comparable sizes and the average domain size is reduced by a factor, roughly equal to 2, from level l to level $l+1$. These properties are inherited from the bottom-up coarsening scheme used by the AMG method. Furthermore, the clustering of the field $v_1 = (-1,1)$ (Fig. 5 a-d) is perfectly aligned with the field (cf. the basis functions in Fig. 4 a-d). However, the clustering of the field $v_2 = (1,1)$, although very similar, is less regular (Fig. 5 e-h). This is the unavoidable impact of the underlying operator discretization (mesh). Since v_2 is perpendicular to the initial mesh edges, this is the worst-case scenario. However, even in this case, the constructed domains are still very much aligned with the field. Figure 6 demonstrates this for a 2D circular field. Finally, note that the supports of the basis functions, respectively the induced domains on a given level, do not have *exactly* the

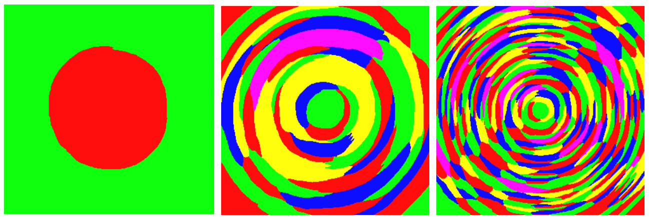

Figure 6: Circular field, levels C, $C-2$, $C-4$ (C=coarsest)

same size (area), since AMG cannot evaluate (integrate) the *mass* of the basis functions. Indeed, AMG does not employ any *geometric* nodal information, but only a matrix of coupling strengths. As discussed in Sec. 6, these restrictions cause no practical problems for visualizing real-world datasets.

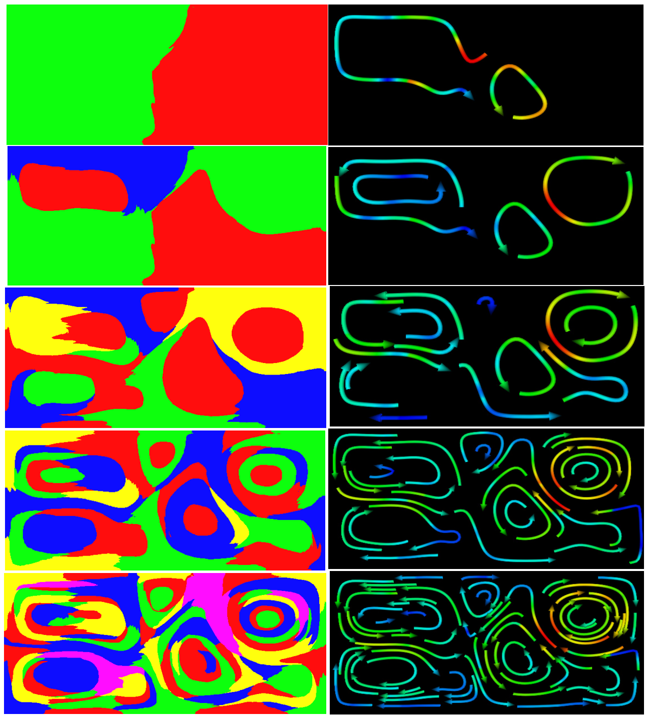

Figure 7: Convective flow decomposition, five coarsest levels (top to bottom). Left: domains. Right: arrow icons colored by vector magnitude

6 Applications

6.1 2D flow fields

We first consider a convective, incompressible flow field on a two-dimensional domain (Fig. 7). Heating from above and cooling from below result in convective patterns. The flow is modelled by the incompressible Navier–Stokes equations using the Boussinesq approximation. Figure 7 shows convective patterns of the resulting flow at successively finer levels of detail. Besides color-coded domains, we visualize the multiscale by velocity-colored curved arrow icons (cf. [21, 11]). For every domain $\mathscr{D}^{l,i}$, we draw one such icon, using a streamline seeded at the point where the corresponding basis $\Psi^{l,i}$ is maximal. By construction (cf. Sec. 5) this point stays inside the respective domain $\mathscr{D}^{l,i}$ regardless of the domain's shape. Other choices for the seed point, or icon, are also possible. Figure 10 shows the decomposition of a magneto-hydrodynamics (MHD) flow dataset, as a second 2D example.

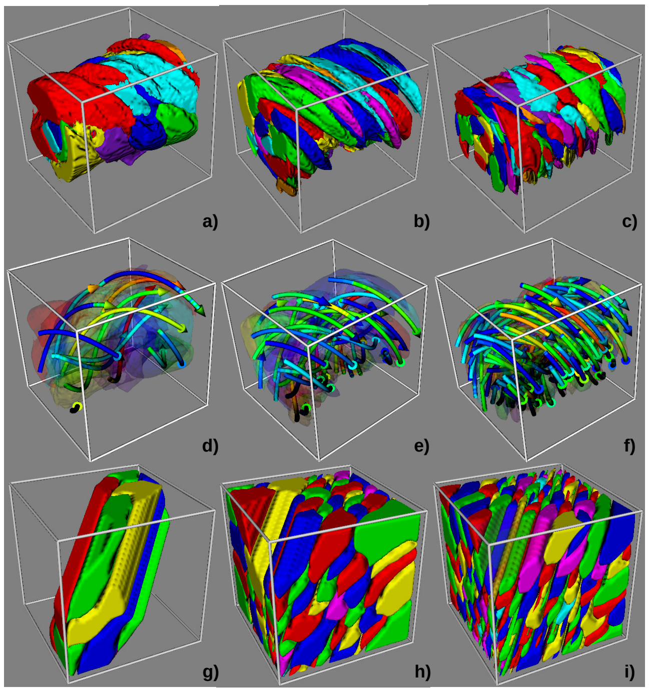

Figure 8: Helix flow, selected domains (a-c), half-transparent domains with arrow icons (d-f). Diagonal flow, selected domains (g-i).

6.2 3D flow fields

Our method works identically for 3D (volumetric) vector fields. The only difference is the use of tetrahedral, instead of triangular, meshes. However, direct visualization of a color-coded domain decomposition, as in the 2D case, is not effective due to the volumetric occlusion. Hence, we use a few postprocessing steps. For every domain $\mathscr{D}^{l,i}$ on a given level l, we construct a closed triangle mesh that bounds $\mathscr{D}^{l,i}$. Next, we relax (smooth) these meshes using e.g. a Laplacian filter or a windowed sync filter [19]. As a result, the meshes become slightly smaller, which allows us to better separate them visually. Next, we implement an interactive navigation scheme in which domains $\mathscr{D}^{l,i}$ can be made half or completely transparent by a mouse click. Users can interactively 'carve' into the flow volume to e.g. remove uninteresting areas and bringing inner flow structures into sight, see Fig. 8. Alternatively, we can visualize the flow at a given level of detail using the same colored arrow glyphs as in the 2D case.

Figure 8 a-f,g-i shows the first three coarsest decomposition levels of a 3D helix flow and of a 3D laminar flow with $v = (1,1,1)$ respectively. We used the interactive technique sketched above to remove the outer domains and to expose the more interesting inner flow structure. The remaining smoothed domains are shown in Fig. 8 a-c for the helix flow and Fig. 8 g-i for the laminar flow (compare the latter with the 2D field in Fig. 5). Figures 8 d-f show the same domains as Fig. 8 a-c, this time half transparent and with the same arrow icon technique as in 2D (Sec. 6.1). Finally, we consider the incompressible flow in a water basin with two interior walls, an inlet (on the left) and an outlet (on the right), the same dataset as in [9]. Figure 9 shows several multiscale levels, visualized with curved arrow icons. These images show that our method scheme works in 3D just as well as in 2D.

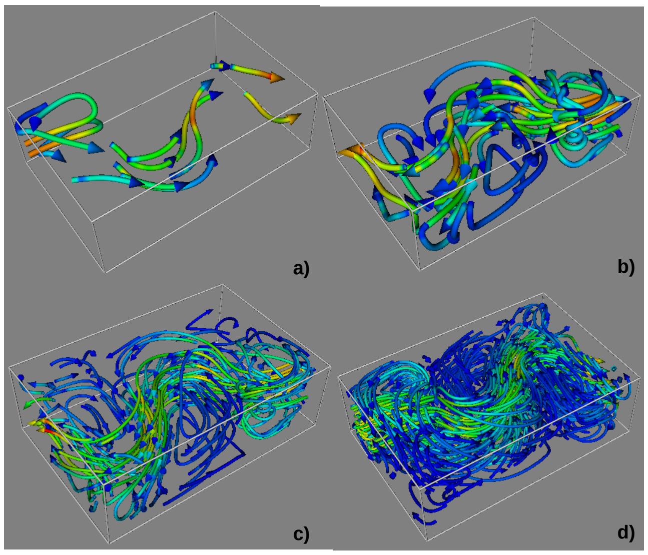

Figure 9: Water basin, four coarsest levels (a-d)

6.3 Flow fields on surfaces

So far we considered vector fields on domains which are subsets of 2D and 3D Euclidean space. We now extend our method to cluster tangential flow fields on surfaces, such as weather-map wind fields over the Earth, flow fields on streamsurfaces, or vector fields in differential geometry. To this end, we have to generalize the differential operator $\mathscr{A}$. Let us consider a smooth manifold $\mathscr{M}$ embedded in $\mathbb{R}^3$. On $\mathscr{M}$ we can define a gradient of a smooth function u as follows: First we extend u to a neighbourhood of $\mathscr{M}$, then we compute the Euclidean gradient in $\mathbb{R}^3$. Finally, we project this gradient onto the tangent space $\mathscr{T}_x\mathscr{M}$. Overall, this procedure leads to the definition

$$\nabla_{\mathscr{M}} u := P_{\mathscr{T}_x\mathscr{M}} \nabla_{\mathbb{R}^3} u\,,$$

where u is assumed to be already extended to $\mathbb{R}^3$ and $P_{\mathscr{T}_x\mathscr{M}}$ denotes the projection onto the tangent space $\mathscr{T}_x\mathscr{M}$. Furthermore, we define the divergence $\mathrm{div}_{\mathscr{M}} v$ for a vector field v on the tangent bundle as the dual operator of the gradient by

$$\int_{\mathscr{M}} \mathrm{div}_{\mathscr{M}} v\,\phi\,\mathrm{d}A := -\int_{\mathscr{M}} v \cdot \nabla_{\mathscr{M}} \phi\,\mathrm{d}A$$

for all smooth, compactly supported functions ϕ on $\mathscr{M}$. With these operators at hand, we obtain the analogue of the differential operator $\mathscr{A}$ on an Euclidean domain:

$$\mathscr{A} u := -\mathrm{div}_{\mathscr{M}} (A \nabla_{\mathscr{M}} u)$$

for C^2 functions u in $\mathscr{M}$. The finite element discretization is now completely analogous to the above Euclidean case. In fact, we use exactly the same code for all our applications. We approximate the surface $\mathscr{M}$ by a triangulation $\mathscr{M}_h$ and obtain the stiffness matrix

$$A_{ij} := \int_{\mathscr{M}_h} a(v)\,\nabla\Phi^i \cdot \nabla\Phi^j\,\mathrm{d}A\,,$$

where Φ^i, $i = 1,\cdots n$, are the piecewise linear nodal basis functions of the triangulation $\mathscr{M}_h$, as in Sec. 4.

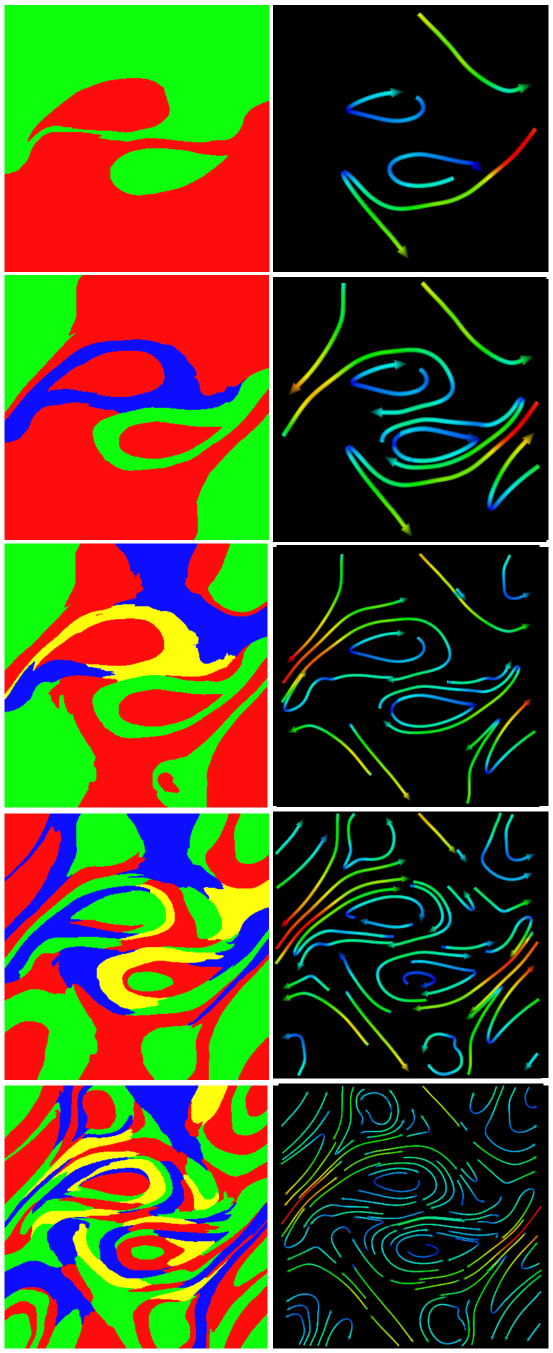

Figure 10: MHD flow decomposition

As an illustration, we give the multiscale decomposition of the average wind stress field on the surface of the Earth in Fig. 11 (the dataset is taken from [28]). The flow texture in the bottom row was produced with the IBFV method described in [28].

7 LOCAL LEVEL OF DETAIL

Our multiscale method presented so far displays a flow field at a user-selected *global* level of detail. However, it is often desired to have a spatially varying level of detail that allows for a user-selected focus on certain flow regions.

We provide this in an interactive, flexible and intuitive way using our multiscale decomposition, as follows: Figures 12 a,b illustrate the procedure for a Benard incompressible 2D flow. First, we select a *point of interest* with a mouse click. We next display a scale of domains $\{\mathscr{D}^{k,p}\}$, ranging from fine to coarse levels, which all contain the selected point. These domains are shown in a nested fashion with different colors (finest domain is red, see colormap in Fig. 12 a). A second mouse click selects an *area of interest*, i.e., a particular domain $\mathscr{D}^{l,i}$ on a level l from this scale. A third click on one of the finer domains $\mathscr{D}^{m,j}$ from the scale $\{\mathscr{D}^{k,p}\}$ chooses the *level of detail* $m \leq l$. Finally, we cover the domain $\mathscr{D}^{l,i}$ with flow icons corresponding to all finer domains $\mathscr{D}^{m,j}$ which intersect $\mathscr{D}^{l,i}$. The result of a four-fold application (12 clicks) of this procedure is depicted in Figure 12 b. Here, we selected the four vortices of the flow domain for display. For better orientation, we also show all domains on the finest level in the gray background. In Figure 12 c, we show several basis functions corresponding to domains selected by this procedure, this time for the Boussinesq dataset (Fig. 7). Finally, Figure 12 d shows three regions of interest selected with 9 clicks for the Earth dataset, one above Europe and two above Africa. This strategy can be also implemented for 3D datasets in a straightforward way.

8 Discussion

The presented results clearly show that our approach for multiscale representation of the structure of a vector field works robustly and generically for 2D, curved surfaces, and 3D datasets. The main ingredients of our method are the use of a differential operator $\mathscr{A}$ which encodes the flow field, and an AMG method which computes a multiscale of basis functions based on a given fine level discretization of $\mathscr{A}$. A single set of parameters performed well for all considered datasets.

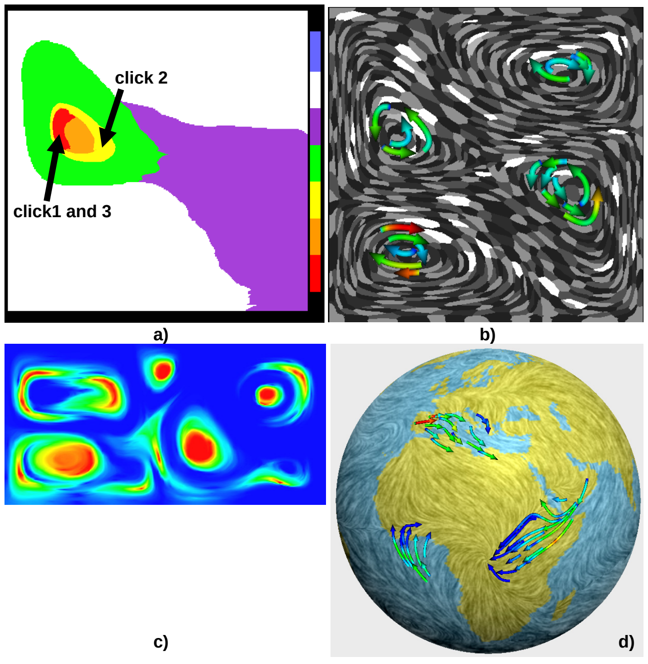

Figure 12: Local level of detail. Selection procedure (a), result after four selections (b). Selected basis functions on Boussinesq dataset (c). Three selections on Earth dataset (d)

Two aspects of our method must be emphasized again. First, our AMG-based coarsening works purely bottom-up. First, clustering decisions taken on a fine level cannot be reversed on a higher level. Secondly, AMG operates on matrix data only. It has no access to geometric information, as mentioned in Sec. 5. These two aspects may lead to coarse levels where cluster sizes can be unbalanced and vector field symmetries may not be exactly captured. These issues, however, are encountered in many bottom-up clustering methods. If desired, they can be corrected in an additional postprocessing phase, e.g. by rebalancing the cluster masses via additive and multiplicative biasing of the defining basis functions (Sec. 5).

An important feature of our method is its speed and ability to treat very large datsets. Table 1 shows, for several datasets, the time and memory needed for the matrix assembly and the time needed by the AMG multiscale computation on a Pentium 4 at 1.8 GHz with 1MB RAM running Linux. The actual computation of the domain decomposition (Sec. 5) involves just the recursive multipication of all prolongation matrices. This is done on-the-fly, as the user visually navigates through the multiscale. The run time of AMG per node is roughly five times larger for 3D datasets than for 2D datasets. This factor is exactly the ratio of the sparsities of the corresponding coupling matrices, i.e. the ratio of the number of nodal neighbors in 3D tetrahedral and 2D triangular meshes respectively. Using 3D hexahedral meshes only increases the time and memory requirements, since the average node connectivity in such meshes is larger. Moreover, our analysis of the results showed that hexahedral meshes do no give any advantage with respect to the decomposition quality. Finally, we remark that practically all the effort is spent in computations involving matrix-vector products. Consequently, classical speedup strategies, e.g. parallelization and/or graphics hardware acceleration, can be directly applied.

Dataset	Dimension (nodes)	Assembly time	AMG time	AMG speed μsecs/node	Matrix (KB)
Laminar 2D	200x200	0.81	1.49	37	4.9
Circle	100x100	0.30	0.32	32	1.2
Benard	256x256	1.01	2.41	36	8.1
MHD	256x256	1.00	2.40	36	8.1
Boussinesq	256x512	2.05	4.62	35	16.2
Earth	12844	0.40	0.37	29	1.6
Laminar 3D	30x30x30	0.91	4.12	152	3.4
Helix	50x50x50	4.27	19.83	158	15.6
Waterbox	128x85x42	13.75	90.73	198	57.1

Table 1: Algorithm timings

9 Conclusions

We have presented a novel method for the multiscale representation and visualization of vector fields. Our method is based on an encoding of the flow field in a differential operator. The discretization of this operator, essentially a stiffness matrix which encodes the flow, is transformed into a multiscale representation by the algebraic multigrid (AMG) method. We then use the associated multiscale of basis functions to decompose the flow domain in flow-aligned regions. Finally, this decomposition can be visualized e.g. by classical flow icons. The most attractive aspect of our method is probably its generality: We are able to treat flow fields on 2D flat and 3D curved surfaces and 3D volumes with the *same* method using the same implementation and parameter settings.

In the future, we plan to generalize our approach to other types of flow operators, based on different coupling strength definitions (e.g. using vorticity), which will result in different flow simplifications. Furthermore, the multiscale of basis functions constructed by the AMG can be used to design different visualization techniques as well (cf. Fig. 12 c). One idea is to use them as color and/or transparency information in texture advection based methods [27, 20] to produce multiscale textured animations of flow fields.

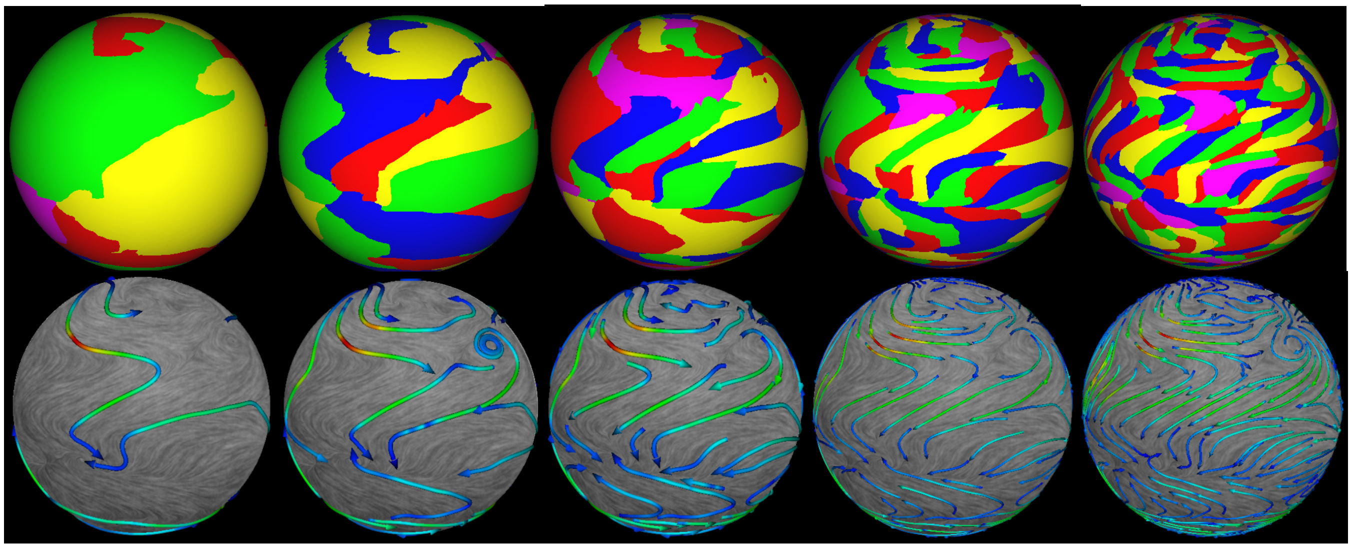

Figure 11: Climate dataset decomposition, five coarsest levels (left to right). Domains (top row) and flow texture overlaid with curved arrow icons (bottom row)

Acknowledgement. We thank E. Baensch, A. Dedner and M. Wesenberg for providing the 2D and 3D data sets. We thank the DFG for the support of T. Preusser via the special research program on time series analysis and image processing. Finally, we also acknowledge the support of the Sonderforschungsbereich 611.

References

[1] D. Bauer and R. Peikert. Vortex tracking in scale space. *Proc. Visualization'02, pp. 233-241*, 2002.

[2] A. Brandt. Algebraic Multigrid Theory: The Symmetric Case. In *Preliminary Procs. of the Intl. Multigrid Conf.*, Copper Mountain, Colorado, April 1983.

[3] A. Brandt. Algebraic Multigrid Theory: The Symmetric Case. *Appl. Math. Comput.*, 19:23–56, 1986.

[4] A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic Multigrid for Automatic Multigrid Solutions with Application to Geodetic Computations. Technical Report, Institute for Computational Studies, Fort Collins, Colorado, October 1982.

[5] A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic Multigrid for Sparse Matrix Equations. In D. J. Evans, editor, *Sparsity and Its Applications*. Cambridge University Press, 1984.

[6] M. Brezina, A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, and J. W. Ruge. Algebraic Multigrid Based on Element Interpolation (AMGe). *SIAM J. Sci. Comp.*, 22(5):1570–1592, 2000.

[7] T.P. Chartier. Spectral AMGe (ρAMGe). *Abstracts of the Seventh Copper Mountain Conference on Iterative Methods*, 2, 2002.

[8] U. Clarenz, M. Griebel, M. Rumpf, M. A. Schweitzer, and A. Telea. Feature sensitive multiscale editing on surfaces. *to appear in The Visual Computer*, 2004. numerik.math.uni-duisburg.de/research/publications.htm.

[9] U. Diewald, T. Preußer, and M. Rumpf. Anisotropic diffusion in vector field visualization on euclidean domains and surfaces. *IEEE TVCG*, 6(2):139–149, 2000.

[10] J. Ebling and G. Scheuermann. Clifford convolution and pattern matching on vector fields. *Proc. Visualization'03, pp. 193-200*, 2003.

[11] H. Garcke, T. Preusser, M. Rumpf, A. Telea, U. Weikard, and J. J. van Wijk. A phase field model for continuous clustering on vector fields. *IEEE TVCG*, 7(3):230–241, 2000.

[12] T. Grauschopf, M. Griebel, and H. Regler. Additive Multilevel-Preconditioners based on Bilinear Interpolation, Matrix Dependent Geometric Coarsening and Algebraic Multigrid Coarsening for Second Order Elliptic PDEs. *Appl. Num. Math.*, 23(1):63–96, 1997.

[13] M. Griebel, D. Oeltz, and M. A. Schweitzer. An Algebraic Multigrid Method for Linear Elasticity. *SIAM J. Sci. Comp.*, 25(2):385–407, 2003.

[14] B. Heckel, G. Weber, B. Hamann, and K. I. Joy. Construction of vector field hierarchies. *Proc. Visualization '99, IEEE Computer Society Press, pp. 19-25*, 1999.

[15] A. Jain, M. Murty, and P. Flynn. Data clustering: a review. *ACM Computer Surveys*, 31(3):264–323, 1999.

[16] B. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. *Visualization in Scientific Computing '97*, 1997.

[17] K. Polthier and E. Preuss. Variational approach to vector field decomposition. *Proc. VisSym'00, pp. 147-156*, 2000.

[18] J. W. Ruge and K. Stüben. Efficient Solution of Finite Difference and Finite Element Equations by Algebraic Multigrid. In D. J. Paddon and H. Holstein, editors, *Multigrid Methods for Integral and Differntial Equations*, The Institute of Mathematics and its Applications Conference Series. Clarendon Press, 1985.

[19] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Prentice Hall, 1995.

[20] A. Telea and J. J. van Wijk. *3D IBFV*: Hardware-accelerated 3d flow visualization. *Proc. Visualization '03, pp. 233-240*, 2003.

[21] A. C. Telea and J. J. van Wijk. Simplified representation of vector fields. *Proc. Visualization '99, IEEE CS Press, pp. 35-42*, 1999.

[22] H. Theisel and H. P. Seidel. Feature flow fields. *Proc. EG/IEEE VisSym'03, pp. 141-148*, 2003.

[23] Y. Tong, S. Lombeyda, A. Hirani, and M. Desbrun. Discrete multiscale vector field decomposition. *ACM TOG, 22(3)*, pages 445–452, 2003.

[24] X. Tricoche, G. Scheuermann, and H. Hagen. Continuous simplification of planar vector fields. *Proc. Visualization, pp. 159-166*, 2001.

[25] U. Trottenberg, C. W. Osterlee, and A. Schüller. *Multigrid*, Appendix A: An Introduction to Algebraic Multigrid by K. Stüben, pages 413–532. Academic Press, San Diego, 2001.

[26] G. Turk and D. Banks. Image-guided streamline placement. In *Proc. SIGGRAPH*, pages 453–460, 1996.

[27] J. J. van Wijk. Image based flow visualization. *Computer Graphics (Proc. SIGGRAPH '01), ACM Press, pp. 263-279*, 2001.

[28] J. J. van Wijk. Image based flow visualization for curved surfaces. *Proc. Visualization'03*, pages 123–130, 2003.