Wavelet Compression of Vector Field Visualizations

VIRGINIE MARION-POTY, WILFRID LEFER

Université du Littoral Côte d'Opale
B.P. 719, 62228 Calais, France
e-mail: {poty,lefer}@lil.univ-littoral.fr

Abstract: Flow visualization has been an active research field for several years and streamlines have proved to be an effective representation of two-dimensional steady vector fields. Online simulation of complex dynamic systems, computed on powerful computers, become more and more common. Efficient steering of the computational process requires the analysis of simulation results in real time. Visualization offers the possibility of at-a-glance analysis of the data but requires to be able to download visualizations efficiently. Here we propose a wavelet model and compression scheme for streamline based images of 2D steady vector fields. This method allows us to decrease the size of the data by a factor of 2.5 for a visualization quality comparable to the original image, and the compression factor increases to 10 if we accept small deformations.

1 Introduction

Vector field data are produced by scientific experimentations and numerical simulations, which are now widely used to study complex dynamic phenomena, with various areas of applicability, such as global climate modelling and computational fluid dynamics. Large-scale, time-varying simulations are able to produce large amounts of data in a short time and raises the need for effective techniques to get insight in the data and to extract meaningful information.

Several techniques have been proposed to visualize steady flow fields, including icon plots, line representations, and textures. A streamline is a line tangential to the vector field at any point. The construction of a streamline is performed by integrating the trajectory of a massless particle that would have been released in the flow at a given location. By covering the image with a set of streamlines, the global structure of the vector field can be visualized, its degree of turbulence can be estimated, and the position of all critical points, such as sinks and saddle points, can be easily located in the domain.

In order to visualize a vector field properly it is necessary to pay attention to the quality of the visualization. Following several observations we have made, it appears that three main factors have a strong influence on the quality of a streamline-based image:

- the length of the streamlines: the major visual property of a streamline is that it allows the observer to follow the path of the flow visually, thus the longer the streamline the better the quality of the visualization.
- streamline ends, which appear as visual artefacts in the image, hence disturbing the interpretation of the flow features. The tapering effect has proved to be a good way to reduce the visual impact of streamline ends on the quality of the visualization [7].
- image density, which can be defined as the ratio between colored pixels, i.e. pixels covered by a streamline, and background pixels. If the image density is not uniform, regions with a high density tend to appear as more important than the others, because they concentrate more visual features. Unless this is a desired feature, whose aim is to highlight regions of particular interest, it is important to ensure a uniform density. This is illustrated by figure 1.

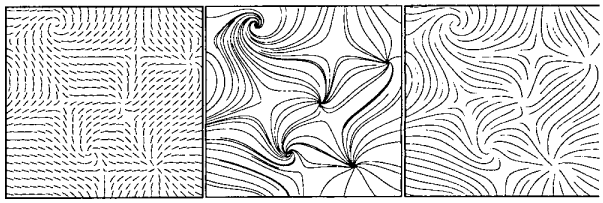


Figure 1: **A two-dimensional steady vector field visualized by means of streamlines.** Left and middle figures have been obtained by placing seed points at the intersections of a regular grid. Left: short streamlines. Middle: long streamlines. Right: image obtained by our streamline placement algorithm.

Turk and Banks proposed an image guided streamline placement algorithm, which selects the best set of streamlines among all possible ones so as to obtain a uniform density in the image [7]. Their method was extended to curvilinear grids later [4]. In [2] we have presented a more effective approach to address the streamline placement issue. In [3] we proposed an extension of this technique to compute streamline images at different *levels of density*. Basically a set of images are computed, each one having a given (uniform) density. Passing from one image to the next, resp. the previous, that is increasing, resp. decreasing, the density, is achieved by just adding, resp. deleting, a few streamlines. In other

1530-1834/01 $10.00 © 2001 IEEE

words the streamlines sets are nested, a streamline defined at level J of the hierarchy being defined for all levels J'>J. Such a representation is very suitable for interactive exploration of vector fields, the density being adjusted on demand without any additional computation.

In this paper we extend the multiresolution property by proposing a wavelet scheme for the representation of streamline. While the previous work did only address multidensity image representation, here we address the multiresolution representation of streamlines themselves. With both techniques we are now able to generate images of a vector field of an arbitrary memory size. Thus we do not only allow the user to monitor the density of the image in real time but we can maintain any frame rate during the exploration of the data by adjusting the level of details of the streamlines representation. Particularly, in the case of remote exploration of large data sets, the level of details can be chosen as a function of the transfer rate.

It is important to note that we are interested in having a geometric representation of the streamlines rather than just an image of the vector field. This does not only allow the progressive transmission of the visualization but we are also able to use the streamlines for various visualization effects, such as mapping textures on them or highlighting some particular flow path by dye advection for instance. Hence traditional image compressing techniques are not suitable in this case. The remaining of this paper is as follows. In section 2 we give the necessary background on the multiresolution theory, which is the mathematical basement of this work, and we describe the computation of a streamline. Section 3 describes our wavelet scheme for streamlines and the results are presented in section 4. Last, section 5 allows us to conclude.

2 Background

Here we give the necessary background to understand the next section. We detail the streamline computation and particularly an important property of our streamline representation: they are discrete curves with evenly spaced points. Then the part of the multiresolution theory that is related to this work is presented.

2.1 Streamlines

A vector field is generally given by a set of points, with various possible organizations, and a vector associated with every point. Hence computing a streamline, that is a flow path, can not be achieved analytically but rather is performed by approximation, using a so-called *integrator*. Such a program typically generates a sequence of ordered points, which are called *sample points*.

The computation of a streamline starts by determining the first sample point, called a *seed point*, from which the integration of subsequent sample points occurs. The position of this first point is important, especially for streamline placement, although it does not have any effect on the quality of the results produced by the method presented here. Once a valid seed point has been found, we process the integration forward and backward from that point, in order to compute the positions of all sample points for the current streamline. Basically each integration step yields a new sample point. There is a wide family of integrators that can be used to integrate flow paths [5]. We use an integrator that produces evenly-spaced sample points, such as DOPRI5 [1]. The construction of a streamline stops when we reach a singularity of the flow (source or sink) or the border of the computational domain has been reached.

To summarize this section, it is important to remember that a streamline is a discrete curve composed of evenly spaced points.

2.2 Multiresolution Theory

Let us introduce wavelets, a well adapted model for the representation of functions at different levels of details (LOD).

2.2.1 Hierarchical Decomposition of Curves

Curves can be seen as piecewise-linear functions. Let V^0 denote the vector space of all functions that are linear over the entire interval $[0,1)$. With this notation, V^j includes all piecewise-linear functions defined on the interval $[0,1)$ with linear pieces over each of 2^j equal-sized subintervals. A piecewise-linear function over two intervals can always be defined as a piecewise-linear function over four intervals, each interval for the first function corresponding to a pair of intervals for the second one. Spaces V^j are said to be nested, which is noted as

$$V^0 \subset V^1 \subset V^2 \subset \dots$$

The basis functions for spaces V^j are called *scaling functions* and are usually denoted by the symbol ϕ. Building a multiresolution analysis imposes the definition of a new vector space W^j as the orthogonal complement of V^j in V^{j+1}. A collection of linearly independent functions $\psi_i^j(x)$ spanning W^j are called *wavelets*.

Then a function $f(x)$ expressed in V^{j+1} as

$$f(x) = c_0^{j+1}\phi_0^{j+1}(x) + c_1^{j+1}\phi_1^{j+1}(x) + \dots + c_{2^{j+1}-1}^{j+1}\phi_{2^{j+1}-1}^{j+1}(x)$$

can be rewritten in terms of basis functions in V^j and W^j as

$$f(x) = c_0^j \phi_0^j(x) + \ldots + c_{2^j-1}^j \phi_{2^j-1}^j(x) + d_0^j \psi_0^j(x) + \ldots$$
$$\ldots + d_{2^j-1}^j \psi_{2^j-1}^j(x)$$

The main idea behind multiresolution analysis is the decomposition of a function into a low-resolution part represented by the coefficients c_i^j and a "detail" part represented by d_i^j. Such a decomposition can be used for compression, as explained in the next section.

2.2.2 Wavelet Compression

Wavelet decomposition is very suitable for efficient compression of functions defined over various domains. Wavelet-based compression is lossy and consists typically in a three steps algorithm:

1) *Filter bank decomposition*: the original function, defined by a sequence c^j, is broken down into a coarse level approximation c^0 and wavelet coefficients d^0, d^1, ..., d^{j-1} corresponding to the levels 0, ..., j-1, respectively.

2) *Selection*: a subset s of the detail coefficients is selected from d^0, d^1, ..., d^{j-1} according to a given criterion.

2) *Reconstruction*: a function is reconstructed from the coarse level approximation c^0 and the selected wavelets coefficients.

The decomposition and reconstruction steps are implemented by applying filters to the coefficients defining the function. We now introduce the concept of filter bank.

2.2.3 The Filter Bank

In [6] Stollnitz et al. proposed the following definition for a filter bank. Consider a function in some approximation space V^j. Let's assume we have the coefficients of this function in terms of scaling function basis. This coefficients can be written as a column matrix of values:

$$c^j = [c_0^j \ldots c_{v(j)-1}^j]^T$$

where c_i^j could be the coordinates of a curve's control points in $\mathbf{R}^2$.

Suppose we wish to create a low-resolution version c^{j-1} of c^j with a smaller number of coefficients $v(j-1)$. The standard approach for creating these values is to use some form of linear filtering and to down-sample on the $v(j)$ entries of c^j. This process can be expressed as a matrix equation:

$$c^{j-1} = A^j c^j$$

where A^j is a constant matrix.

Since c^{j-1} contains fewer entries than c^j, it is intuitively clear that some amount of detail is lost in this filtering process. If A^j is appropriately chosen, it is possible to capture the lost detail as another column matrix d^{j-1}, computed as

$$d^{j-1} = B^j c^j$$

where B^j is a constant matrix related to A^j.

A^j and B^j are called *analysis filters*. The process of splitting the coefficients c^j in a low-resolution part c^{j-1} and detail d^{j-1} is called *analysis* or *decomposition*. Then c^{j-1} can be decomposed into a low-resolution part c^{j-2} and detail d^{j-2}, and so on. Thus, the original coefficients c^j can be expressed as a hierarchy of lower-resolution parts c^0, c^1, ..., c^{j-1} and details d^0, d^1, ..., d^{j-1}. This recursive process is known as a *filter bank*.

Moreover if A^j and B^j are appropriately chosen, the original coefficients c^j can be recovered from c^{j-1} and d^{j-1} by using two matrices P^j and Q^j as follows:

$$c^j = P^j c^{j-1} + Q^j d^{j-1}$$

P^j and Q^j are called *synthesis filters*, and the process of recovering c^j from c^{j-1} and d^{j-1} is called *synthesis* or *reconstruction*.

3 Streamlines Compression

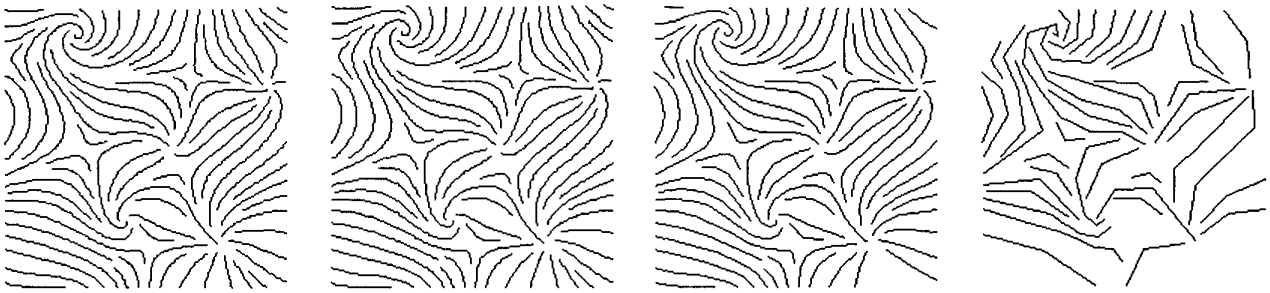
Now we describe our wavelet scheme for streamline and the different steps of the compression.

3.1 Decomposition Step

Stollnitz et al. show in [6] that the only functions that can be hierarchically decomposed are those that can be generated by a *recursive subdivision* process. The idea of a recursive subdivision process is to create a function by repeatedly refining an initial piecewise-constant function $f^0(x)$ to produce a sequence of increasingly detailed functions $f^1(x)$, $f^2(x)$, ..., that converge to a limit function $f(x)$. The function $f^j(x)$ is constructed from function $f^{j-1}(x)$ in 2 steps: the *splitting step*, which introduces midpoints, and the *averaging step*, which computes weighted averages. For all subdivision schemes the splitting step is the same, and only the averaging step differs.

If the function $f^{j-1}(x)$ is characterized by the sequence

$$(c_0^{j-1}, c_1^{j-1}, \ldots, c_{2^{j-1}-1}^{j-1})$$



a. original image | b. compression rate of 50% | c. compression rate of 90% | d. compression rate of 95%

Figure 3: Compression of streamlines images with our wavelet scheme. We are able to compress the data by a factor of 10 without any visual degradation.

then the splitting step for constructing $f^j(x)$ is written as follows:

$$c_{2i}^{j} = c_{i}^{j-1}$$

$$c_{2i+1}^{j} = \frac{1}{2}\left(c_{i}^{j-1} + c_{i+1}^{j-1}\right)$$

The averaging step is of the form:

$$c_{i}^{j} = \sum_{k} r_k c_{i+k}^{j}$$

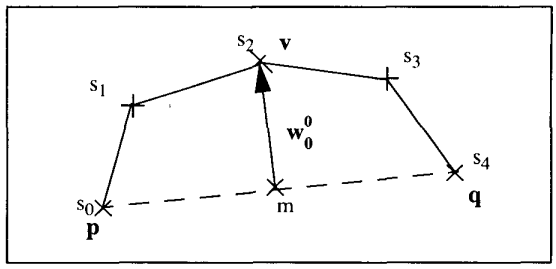
r_k masks for various subdivision schemes are described in [8]. As Lounsbery explained in [8] the chief property of piecewise linear subdivision that leads to simpler algorithms is tight locality. Then the analysis filters A^j and B^j can be generated by simple technique without the need for implementing sparse matrix multiplication. Instead, the wavelet coefficients are determined as:

$$w = v - \frac{1}{2}(p+q)$$

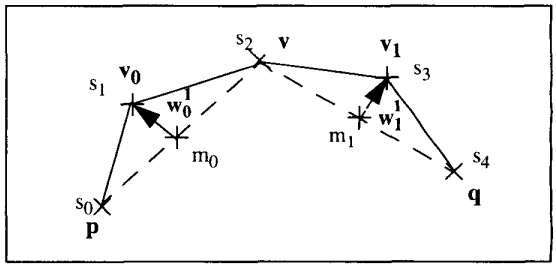
where w is the wavelet coefficient associated with a new vertex v which has been decomposed from the coarser-level vertices p and q. Let us note that this definition of the wavelet coefficient, i.e. the distance between m and v, is a good evaluation of the *approximation error*. Now we propose such a decomposition scheme for streamlines.

Let us recall that a streamline is a discrete curve with *n sample points*. The decomposition process is recursive. It starts by adding point $s_{n/2}$ and computing its distance to the middle-point of points pair (s_0, s_{n-1}). Then this process is repeated with points pairs $(s_0, s_{n/2})$ and $(s_{n/2}, s_{n-1})$, and so forth until all sample points of the streamline have been added. This process is illustrated by figure 2.

Let us remark that this scheme involves that the number of sample points of the former streamline be $2^N + 1$. But it can be easily adapted so as to deal with any streamline: if there is no more sample point between p and q the subdivision process is stopped and no additional point v is generated. The consequence is that the resulting tree will not be well balanced, which is not required by the selection algorithm however.



a. **Computation of level 0:** $p = s_0$, $q = s_4$ **and** $v = s_2$



b. **Computation of level 1: for** w_0^1**:** $p = s_0$, $q = s_2$ **and** $v = s_1$, **for** w_1^1**:** $p = s_2$, $q = s_4$ **and** $v = s_3$

Figure 2: **Decomposition of a streamline.** The sample points of the original streamline are denoted by $s_0, ..., s_{n-1}$. At any step a new point is added as follows: we compute m as the middle point of (p,q) then we compute the wavelet coefficient w as the distance between m and v.

3.2 Hierarchy Construction

All points of a given streamline are organized in a tree whose root corresponds to the coarsest level and subsequent levels correspond to the decomposition levels. Basically, at level 0 of the tree a streamline is defined by only 3 points: the first and last sample points of the former

streamline and the middle sample point. Each streamline being decomposed independently, the resulting structure is a forest composed of all generated trees. This forest is used for both compression and interactive visualization of the vector field.

Information stored at each node includes structural information, for tree traversal, and qualitative information, such as the approximation error. The structure of a node includes:

- the coordinates of the point associated to this node (s_1 on figure 2 for instance),
- the error associated to the node, i.e. the wavelet coefficient (or the distance) associated with that point (w_0^1 for s_1 on figure 2),
- the maximum error of all nodes in its subtree, that is the subtree whose root is the current node. This information will be used for node selection during the reconstruction process.

For the leaves of the trees, the maximum error is obviously the error associated with the node. For the interior nodes, the maximum error is computed by a propagation algorithm from leaves to root.

This is a minimal structure but additional information could be stored here, particularly any information useful for the node selection, such as the degree of interest of the region into which the current point falls.

3.3 Selection and Reconstruction

The forest structure can be used for image compression or interactive visualization. In both cases all the nodes are sorted according to a certain criterion. A usual criterion is the approximation error, the nodes being sorted in decreasing error order. In case of compression a ratio of compression R is defined. Then the K first nodes in the sorted list are selected, where $K = R \times N$, where N is the total number of nodes in the forest. In case of interactive visualization it is important to maintain a certain frame rate, particularly if the data are accessed remotely. Wavelets coefficients are imported in the image continuously, the goal being to have the more accurate visualization as possible at any time. Hence wavelets coefficients are processed in order according to the sorted list.

Generally one try to minimize the approximation error, which is the reason for which we have decided to store the error and the maximum error in each node. For sorting the nodes we use the maximum error because both "parents" of a point should be included in the streamline definition if the point itself is, otherwise unpredictable results can occur, as show on figure 4.

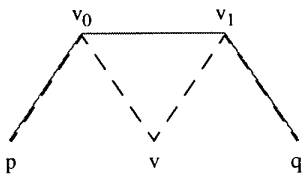


Figure 4: The *maximum error* should be used for sorting wavelets coefficients. Here vertex v has an associated error of 0, while errors for vertices v_0 and v_1 are not zero. If we use the mode *error* for sorting nodes, v_0 and v_1 are used for reconstruction prior to v, which leads to the red curve whereas the original curve is drawn in black. Hence by taking the maximum error this problem is solved.

The selection can be based on various criteria, such as the degree of interest of different parts of the domain or the value of an additional scalar field. If there are several criteria, they are combined in a single function with respective ponderousness coefficients. Then the function is evaluated for each leaf of the forest and the maximum is computed for all other nodes as explained in section 3.2. Then all nodes are sorted according to their values. Note that the hierarchical structure does not have to be updated when the selection criterion changes but only a new sorted list has to be computed.

4 Results and compression ratio

As explained earlier in the introduction, we want to keep the meaning of the streamlines and thus we store for a vector field a list of points associated with it. The average time required to construct such a list on a bi-celeron 466MHz with 128 Mbytes memory is 12.75 µs per point (137261 µs for 10783 points, 56523 µs for 4432 points).

The compression rates are expressed as the ratio between the number of points in the compressed representation and the number of points in the original streamlines.

Table 1: informations about pictures in figure 3

	image a.	image b.	image c.	image d.
number of points	4432	2216	444	222

Table 1: informations about pictures in figure 3

	image a.	image b.	image c.	image d.
compression factor	1	2	10	20
number of streamlines	76	76	71	41
anpps*	58	29	6	5

**anpps* : average number of points per streamline.

The figure 3 and the table 1 show that no significative degradation of the image quality can be observed for compression factors lower than 10. The only degradation we observe is the loss of some "simple" streamlines (streamlines whose shape is close to a straight line). Indeed we see on table 1 that when the compression factor increases, the average number of points per streamline (=*anpps*) decreases. Nevertheless the "complex" streamlines are constructed with more points than anpps and "simple" ones with less points than *anpps*. This tends to unselect "simple" streamlines to keep good informations for "complex" ones.

When the compression rate becomes greater than 90% some streamlines deformations can be observed (see the last image in figure 3).

We can compare these results with those obtained by compressing files containing the lists of points in binary format with lossless compression methods, such as *Unix compress* or *GNUzip*. For the set of streamlines of figure 3, compress and gzip obtain compression rates of 17% and 36%, respectively. Note that such a "binary" compression can also be applied to the compressed result data set from our method.

5 Conclusion

In this paper we proposed a wavelet scheme for the representation of streamline images at different levels of details. This method allows to decompose hierarchically each streamline in a list of points, which can then be globally ordered as a function of various criteria, such as the approximation error or the importance of the data, or a combination of several such criteria. This hierarchical representation of a vector field visualization can be used for compression or progressive transmission. Another crtieria, see in [3], is to maintain a spatial coherence between streamlines like a uniform density over the whole image.

For a visualization quality comparable to the original image, our method achieves higher compression rates than classical non lossy compression algorithms. Moreover our method is suitable for progressive transmission of the data, which is not a feature of the traditional compression methods.

References

1. Hairer, E., S.P. Norsett and G. Wanner. Solving Ordinary Differential Equations I-Nonstiff Problems. Springer Verlag, 1993.
2. Jobard, B. and W. Lefer. Creating Evenly-Spaced Streamlines of Arbitrary Density. In W. Lefer and M. Grave, Editors, Visualization in Scientific Computing'97 (*Proceedings of the 8th Eurographics Workshop on Visualization in Scientific Computing'97*, April 28-30, 1997), pages 43-55, Springer-Wien-New-York.
3. Jobard, B. and W. Lefer. Multiresolution Flow Visualization, poster at the *Int.conf on Graphics, Visualization and Computer Vision 2001*, Plzen, Czech Republic, February 5-9, 2001
4. Mao, X., Y. Hatanaka, H. Higashida and A. Imamiya. Image-Guided Streamline Placement on Curvilinear Grid Surfaces. In D. Ebert, H. Rushmeier and H. Hagen, Editors (*Proceedings of IEEE Visualization'98*, October 18-23, 1998), pages 135-142, IEEE Press.
5. Stalling, D. and H-C. Hege. Fast and Resolution Independant Line Integral Convolution. Computer Graphics Annual Conference Series (*Proceedings of SIGGRAPH'95*, August 6-11, 1995), pages 249-258, ACM Press.
6. Stollnitz, E., T. Derose and D. Salesin. Wavelets for Computer Graphics. Morgan Kaufmann Publishers.
7. Turk, G. and D. Banks. Image-Guided Streamline Placement. Computer Graphics Annual Conference Series (*Proceedings of SIGGRAPH'96*, August 4-9, 1996), pages 453-460, ACM Press.
8. Lounsbery, J.M.. Multiresolution Analysis for Surfaces of Arbitrary Topological Type. Ph.D. in Computer Science and Engineering at University of Washington, 1994.