Hierarchical Streamarrows for the Visualization of Dynamical Systems

Helwig Löffelmann   *Lukas Mroz*   *Eduard Gröller*

Institute of Computer Graphics, Vienna University of Technology

Abstract. Streamarrows are a technique to enhance the use of streamsurfaces by separating arrow-shaped portions from the remaining streamsurface. We present a hierarchical streamarrows algorithm as an extension to this technique: Streamarrows are locally chosen from a stack of scaled streamarrows textures to avoid too big or small streamarrows in the rendered image. We furthermore present techniques how streamarrows can be extended into 3D, namely perpendicular to the streamsurface: streamarrows can be shifted slightly out of the streamsurface. Another extension in this category is to represent the outline of streamarrows as 3D tubes. We show a set of images which have been rendered using this technique and report about ongoing research.
Keywords. Visualization, dynamical systems, streamsurfaces.

1 Introduction

One important area in the field of visualization [10] is the visualization of dynamical systems, e.g., flow fields. Various techniques and algorithms have been developed for such data [4].

A *dynamical system* represents the evolution of a set of variables over time [13]. Real world phenomena as, e.g., the stock market, ecological systems, or chemical reactions are often modeled or simulated as a dynamical system. Assuming a continuous evolution over time, a dynamical system is given as a set of ordinary differential equations (ODEs), which usually cannot be solved analytically. Visualization improves the exploration of such dynamical systems and promotes the understanding of the underlying dynamics.

Typically the temporal evolution of a dynamical system is depicted in *phase space*. Each coordinate of phase space is associated with a system variable. Thus each point in phase space represents a state of the system at a specific moment in time. The local evolution of a state can be expressed by a vector in phase space. This often results in a vector field representation of dynamical systems. The integration of such vector fields starting at a particular point in phase space produces the temporal evolution of the system. This temporal evolution is a curve which is called *trajectory*.

For quite a long time simple techniques like arrow plots or polyline representations of streamlines, streaklines, or pathlines have been used for visualizing dynamical systems [11]. Two-dimensional objects such as streamribbons [16] or streamsurfaces [5][15] are other important techniques for the visualization of dynamical systems. Furthermore extensions to these approaches such as streampolygons and streamtubes [11] or the streamball technique [3] have been important contributions to the visualization of flow fields.

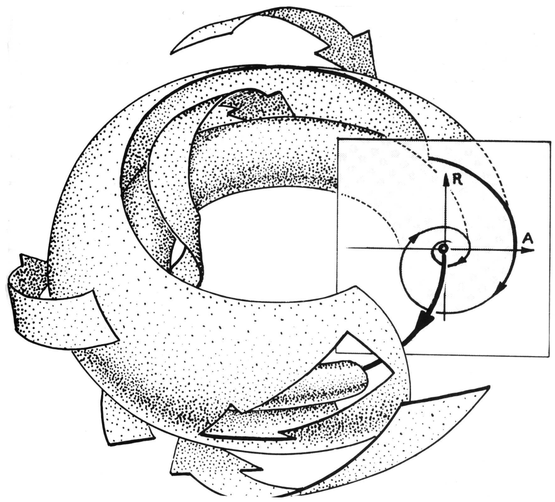


Figure 1: Visualization of a dynamical system by using streamarrows [1]

Although the usage of streamsurfaces often improves the understanding of a system representation, occlusion becomes a problem whenever large opaque surfaces are used for visualization. An interesting technique to deal with this problem is utilized in the book by Abraham and Shaw [1]. They discuss various types of dynamical systems by using hand-drawn illustrations that represent the topological structure of these models. Streamsurfaces make up an important part of many of their images. To reduce the adverse effects caused by occlusion they use arrow-shaped parts of a streamsurface instead of the whole surface. In addition to that they use arrow-shaped holes within streamsurfaces to diminish occlusion. Refer to Figure 1 for a typical image of their book.

Other enhanced methods for the representation of large-scale surfaces have been discussed recently. Interrante et al., for example, derived a technique of representing a semi-transparent surface by using curvature-directed opaque strokes. Their approach is based on an analysis of artistic drawings [6]. In another paper Rheingans proposes an opacity-modulating texture for irregular surfaces as an attempt to reduce the problem of occlusion [12].

2 Streamarrows for mixed-mode oscillations

In cooperation with mathematicians, we investigated a class of dynamical systems that exhibit mixed-mode oscillations [8]. Such dynamical systems are characterized by a periodic pattern of alternating large and small oscillations.

We decided to use streamsurfaces to visualize the model. Due to the particular shape of streamsurfaces resulting from mixed-mode oscillations the resulting occlusion is a major problem. A typical streamsurface for the given dynamical system occludes important parts of itself by forming a shell-like roll with several turns. Self-occlusion is severe from almost any viewing position. Refer to Figure 6a to get an impression of a typical streamsurface within the investigated model. Throughout this project we investigated several techniques for highly occluding streamsurfaces [8]. Most of the methods have been inspired by the hand-drawn images of Shaw and Abraham [1].

The main extension to standard streamsurface techniques was the introduction of streamarrows. A streamsurface is thereby segmented into a set of arrow-shaped objects and the remaining surface portion. By assigning a certain level of semi-transparency to the streamarrows the viewer sees through the “holes” in the streamsurface and gets more information about the structure of the model. In addition to that, the use of streamarrows allows to visualize local information that is available on the streamsurface, e.g., direction of evolution, velocity, and local divergence or convergence (see Figure 6b).

The segmentation of the surface into streamarrows and the remaining surface portions was performed by mapping a regularly tiled texture of arrow-shaped patterns onto the streamsurface and cutting its patches along streamarrows borders. The texture was constructed by specifying a base tile, i.e. the shape of one streamarrow, and tesselating the texture using this base tile. Three sets of geometric objects, namely the inside of streamarrows, the outside, and the separating border, were extracted. After this segmentation either the streamarrows or the remaining surface portions can be assigned a certain level of semi-transparency. See Figure 2 for an illustration of this method.

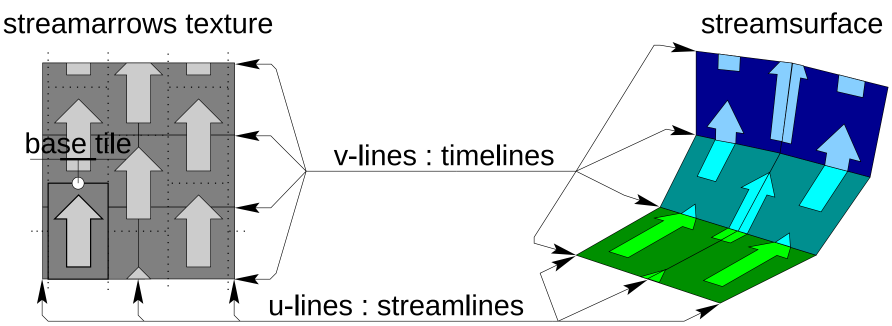


Figure 2: Mapping the streamarrows texture to a streamsurface

A segmentation of a streamsurface into an entirely opaque portion and highly transparent holes (streamarrows) gives a good impression of the interior structure of a curved streamsurface while still retaining a good overview of the spatial arrangement of the streamsurface itself. Using homogeneously transparent surfaces would produce several layers of overlapping streamsurface segments which are quite difficult to interpret spatially.

3 Hierarchical streamarrows

The streamarrows approach [8] is based on a regular tiling of texture space. This is not well-suited for streamsurfaces that spread over regions of high divergence or convergence. The arrows become either too big or too small in certain areas. To eliminate this undesirable effect we present a hierarchical extension to the streamarrows technique.

3.1 A stack of streamarrows textures

The goal of our project was to develop an algorithm, which is capable of generating streamarrows that are almost equal-sized in the rendered image. Since we did not want to loose the ability to represent local divergence or convergence using the streamarrows technique, we decided to construct a hierarchical algorithm instead of a continuous solution. When flow, for example, diverges locally, our method switches discretely to the next detailed level of streamarrows. These streamarrows are smaller in texture space, but since they are used for divergent areas of the streamsurface, they are finally almost equal-sized to all other streamarrows in the rendered image. Refer to Figure 7 for an image which demonstrates this situation.

The *hierarchical streamarrows texture* is specified by the shape of one tile, i.e. the outline of an arrow, and two vectors dc and dr (see Figure 3) which define the offsets between adjacent columns and rows of arrows, respectively. Additionally there is a factor a which represents the scale relation

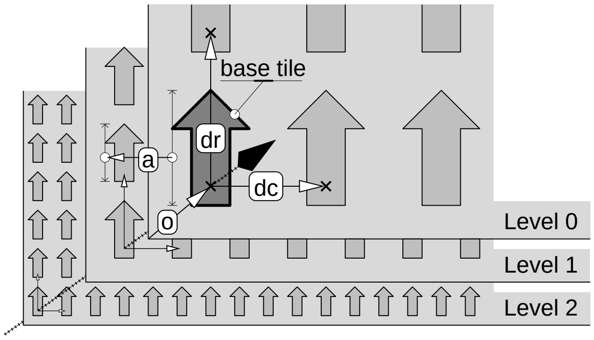


Figure 3: Hierarchical streamarrows texture, specification parameters

between level i and $i + 1$. If a=$1/2$, for example, the size of streamarrows is doubled, when the algorithm switches to the next coarser level. Finally there is a vector o, which is the offset of the entire texture with respect to the origin of texture space. Offset vector o becomes important, when animation is applied. Due to this specification each streamarrow can be addressed by exactly one ID given by three numbers. ID ($level$, col, row) identifies one streamarrow as a copy of the base tile, first translated by o+(dc $\cdot$ col, dr $\cdot$ row), and then scaled about the origin by a^{level}.

3.2 The separation algorithm

The new streamarrows technique itself is triangle oriented, because we produce our streamsurfaces as triangular meshes: the front of the streamsurface ($\approx$ timeline) is advanced through phase space while the streamsurface is integrated. Triangles are smaller in streamsurface areas, where curvature is high, flat areas of streamsurfaces are triangulated with relatively big triangles.

During the streamsurface algorithm vertices are assigned 2D texture coordinates. One coordinate of each vertex is set to the integration time of a streamline from the seed locus to the vertex. The other coordinate of each vertex is set in such a way that all vertices connected with one streamline get the same texture value, i.e. the 1D seed parameter of the start-point of the streamline.

To apply the hierarchical streamarrows texture to the streamsurface the hierarchical streamarrows algorithm processes the streamsurface triangle by triangle and performs the following separation algorithm:

```
activeTiles = {}                          // . . . . IDs of active tiles
lockedTiles = {}                          // . . . . IDs of locked tiles
FOR ALL Triangles tri DO:
| level:=findLevelOfTriangle(tri)         // . get most approprate level
| tiles:=getMaybeTiles(tri,level)         // . . . get overlapping tiles
| FOR ALL Tiles tile IN tiles DO:
| | IF NOT (tile.active OR tile.locked) THEN:
| | | IF overlap(tile,activeTiles) THEN: tile.lock
| | | ELSE:                              tile.activate
| intersect(tri,activeTiles)              // . . . . . do the separation
```

The algorithm needs two data structures in addition to the triangular mesh of the streamsurface: `activeTiles` stores the IDs of all tiles that actually are instanced within the streamsurface, whereas in `lockedTiles` all IDs of tiles are stored that overlap at least one active tile and thus should not be generated. Both data structures need to be be searched as fast as possible (in `overlap()` and `intersect()`) and easy to extend by a new tile. Since tiles are related via their 2D location in texture space and searching is also performed in a 'geographical' manner using texture coordinates (IDs), we use a linked data structure that closely represents the spatial relation between tiles.

The algorithm processes the streamsurface triangle by triangle. For each triangle `tri` the corresponding `level` in the hierarchical streamarrows texture is determined by comparing the size of triangle `tri` in texture space to its size in phase space coordinates. This ratio is used to find the most appropriate level in the stack of streamarrows textures. Then all tiles in that level, which might intersect triangle `tri` are determined (`getMaybeTiles()`). Tiles which are already activated or locked are omitted. All remaining tiles are checked, whether they overlap any active tile (`overlap()`). Tiles that overlap at least one active tile are locked (added to `lockedTiles`), and all the others are activated (added to `activeTiles`). See Figure 4 for an example with two triangles. After all tiles are checked triangle `tri` is intersected with all active tiles and separated

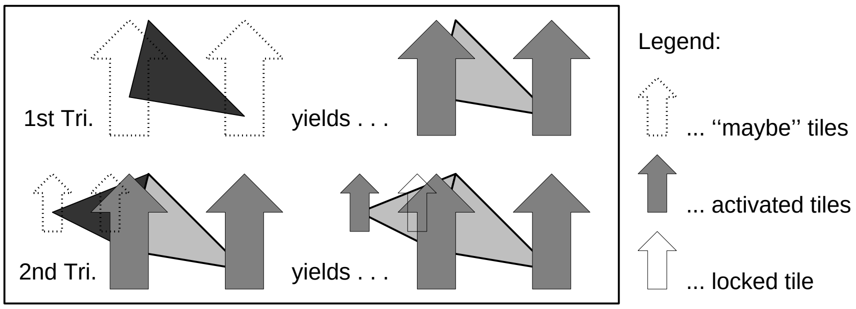


Figure 4: Activating and locking tiles, example with two triangles

into three sets: parts that belong to the arrows, parts that do not, and the separating outline. All three sets can be individually processed, e.g., assigned a certain level of semi-transparency, after the segmentation algorithm has finished.

The mechanism of `activeTiles` and `lockedTiles` ensures that neighboring triangles of a streamsurface are consistently covered by entire tiles of the hierarchical streamarrows texture. If the tile instanced for the currently processed triangle is overlapped by a tile which was already instanced for a previously processed and nearby triangle—this tile is active and of different level than the current tile—the acitve tile is used for the current triangle. This ensures consistency. The current tile which overlaps the active tile is furthermore locked.

4 Working with hierarchical streamarrows

After the separation of streamarrows has been performed (see section 3) various extensions can be applied. For example, two possibilities of extending streamarrows into 3D have been realized.

One possibility is to shift the separated streamarrows slightly in a direction perpendicular to the remaining streamsurface portions. To avoid any confusion—one could interpret the streamarrows as local solutions of the dynamical system, which is certainly not the case in most situations—the shifted streamarrows are connected with the remaining parts of the streamsurface by semi-transparent patches. 3D arrows are generated by this technique, which improves the perception of spatial location and orientation of the streamsurface (see Figure 8a). In this image streamarrows are furthermore enhanced by the use of anisotropic spot noise [14] showing stream- and timelines. Another 3D extension is the representation of the separating outline as 3D tube (see Figure 8b). Using this approach streamarrows can be realized without removing any part of the streamsurface.

Nevertheless almost all of the advantages of the streamarrows technique are preserved as, e.g., indication of flow direction and local velocity.

Streamarrows are implemented in the scope of our DynSys3D system [7], which itself is developed on the basis of AVS [2]. Therefore animation is easily supported. AVS allows to export certain module parameters as input ports. Furthermore there are already some basic animation modules available as, e.g., *animated integer* or *animated float*. Connecting such a module to an exported module parameter is the simplest method of animating a visualization setup.

In our case several parameters can be animated to improve the visualization. Two examples are given: an animated offset vector o (see section 3.1 and Figure 3) can be used to move streamarrows along streamlines. Animating scaling factor a may be used to smoothly decrease the difference between arrows sizes throughout the visualization.

Our streamsurfaces technique has been enhanced by the capability of color coding local attributes as, e.g., velocity magnitude, integration time, divergence, or helicity [9]. This extension allows to focus the viewer's concentration on streamsurface regions that exhibit certain values of a local parameter.

5 Current developments

The project on streamarrows is still work in progress. Some of the ideas concerning this technique are currently under development. They are shortly discussed in this section.

3D Streamarrows can be seen as glyphs with a set of more or less free parameters. Scalar data such as velocity and helicity as well as vector data, for example, vorticity, may be mapped to a streamarrows parameter. We currently investigate the 3D shape attributes of streamarrows as glyph parameters, but also size, 2D shape, opacity, and color.

Up to now we can shift streamarrows slightly out of the streamsurface. This is quite an improvement in certain situations, but has its disadvantages in other cases. Especially if nearby streamsurfaces are not parallel at all—one streamsurface as a stable system solution can be imagined as such a case (see Figure 5)—this technique might yield wrong impressions. We therefore investigate

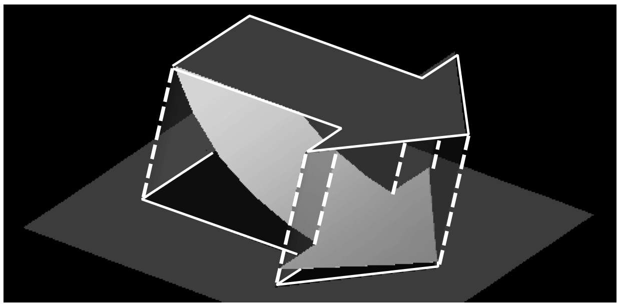


Figure 5: Misleading shifted streamarrows (bright: integrated streamarrow)

another extension, which will allow to build up the shifted streamarrow as local solution by itself. Practically this is achieved by starting a new streamsurface at the shifted tail of our streamarrow and cutting the arrow out of this streamsurface. This technique will allow to represent the dynamical system not just at the streamsurface, but also in its vicinity. Other shapes of 3D streamarrows, e.g., a tent-like shape or variations of the 2D base shape, are also investigated.

Related to the hierarchical streamarrows texture we work on some other extensions concerning the placement of the streamarrows. One idea is to randomly position streamarrows on streamsurfaces. Another idea is to use local streamsurface attributes for the location of streamarrows. For example, surface curvature could be used for this purpose, since this has already been shown to be useful [6]. Finally we think of including viewing parameters to determine optimal placements of streamarrows. This idea was again inspired by the book of Abraham and Shaw [1], where these parameters are included in the hand-drawn illustrations as well.

6 Summary

Streamarrows are a useful technique to enhance the visualization of dynamical systems, i.e. flow fields. In this paper we discuss hierarchical streamarrows. A hierarchical texture is defined for streamsurfaces, which is given as a stack of different sized texture levels. Each level is specified as a set of arrow-shaped tiles that cover the texture of a given level in a regular manner. Controlling parameters of this streamarrows texture allows to improve the use of streamsurfaces by several means:

- Assigning a certain amount of semi-transparency to either the streamarrows or the remaining surface portions allows to reduce the problem of occlusion, which usually occurs when large scale surfaces are used for visualization.
- Local properties of the dynamical system are implicitly visualized by the use of streamarrows. Local convergence or divergence as well as flow velocity are directly represented by the shape of the arrows. Streamlines and timelines within the streamsurface are also visualized using this technique.
- The hierarchical approach allows to use the streamarrows technique also for ill-shaped streamsurfaces. Depending on the local area relation between texture space and phase space, the most appropriate level within the stack of streamarrows textures is chosen for streamarrows instantiation such that finally all streamarrows are more or less of equal size.
- Extensions into 3D furthermore enhance the streamarrows technique. Three-dimensional streamarrows can be modeled by shifting the separated streamarrows slightly above the streamsurface and connecting them with the remaining surface portions by the use of semi-transparent patches. If no parts of the streamsurface should be removed, the arrow outlines can be modeled as 3D tubes.

Currently the following concepts are integrated into our system:

- Interpreting a streamarrow as a glyph allows to further extend this technique. Local attributes of the dynamical system as, e.g., helicity, can be mapped to glyph parameters and thus visualized.
- Using 3D streamarrows it is possible to visualize the vicinity of a streamsurface. Convergence or divergence perpendicular to the streamsurface will thus be visualized.
- Arrow placement is another topic of current research. Breaking the regular distribution of streamarrows within a level allows to selectively place arrows at places of special interest.

Streamarrows are a powerful technique for the visualization of dynamical systems. Streamarrows and hierarchical streamarrows enhance the use of streamsurfaces. These approaches allow an easier interpretation of complex, e.g., curly streamsurfaces. Streamarrows are of limited use for highly convergent or divergent streamsurfaces. For such cases hierarchical streamarrows as proposed in this paper are a better choice. Comparing images produced with the streamarrows technique, e.g., Figure 9, to those in the book of Abraham and Shaw [1], for example, Figure 1, shows that this approach is capable of generating results that are similar to impressive hand-drawings of an artist.

References

[1] R. H. Abraham and C. D. Shaw. *Dynamics – The Geometry of Behavior*. Addison-Wesley, 1992.

[2] Advanced Visualization System Inc. *AVS Developers Guide, Release 4*, May 1992.

[3] M. Brill, W. Djatschin, M. Hagen, S. V. Klimenko, and H.-C. Rodrian. Streamball techniques for flow visualization. In *Proceedings Visualization '94*, pages 225–231, October 1994.

[4] E. Gröller. Application of visualization techniques to complex and chaotic dynamical systems. In M. Göbel and H. Müller, editors, *Visualization in Scientific Computing*, pages 63–71. Springer, 1995.

[5] J. P. M. Hultquist. Constructing stream surfaces in steady 3d vector fields. In *Proceedings Visualization '92*, pages 171–177, October 1992.

[6] V. Interrante, H. Fuchs, and S. Pizer. Illustrating transparent surfaces with curvature-directed strokes. In *Proceedings Visualisation '96*, pages 211–218, 1996.

[7] H. Löffelmann and E. Gröller. DynSys3D: A workbench for developing advanced visualization techniques in the field of three-dimensional dynamical systems. In N. Thalmann and V. Skala, editors, *Proceedings of The Fifth International Conference in Central Europe on Computer Graphics and Visualization '97 (WSCG '97)*, pages 301–310, Plzen, Czech Republic, February 1997.

[8] H. Löffelmann, L. Mroz, E. Gröller, and W. Purgathofer. Streamarrows: Enhancing the use of streamsurfaces for the visualization of dynamical systems. Submitted to the Journal Visual Computer, July 1996.

[9] H. Löffelmann, Z. Szalavári, and E. Gröller. Local analysis of dynamical systems – concepts and interpretation. In N. Thalmann and V. Skala, editors, *Proceedings of The Fourth International Conference in Central Europe on Computer Graphics and Visualization '96 (WSCG '96)*, pages 170–180, Plzen, Czech Republic, February 1996.

[10] G. M. Nielson and B. Shriver. *Visualization in Scientific Computing*. IEEE Computer Society Press, 1990.

[11] F. H. Post and J. J. van Wijk. Visual representation of vector fields: Recent developments and research directions. In Rosenblum et al., editor, *Scientific Visualization – Advances and Challenges*, pages 367–390. Academic Press, 1994.

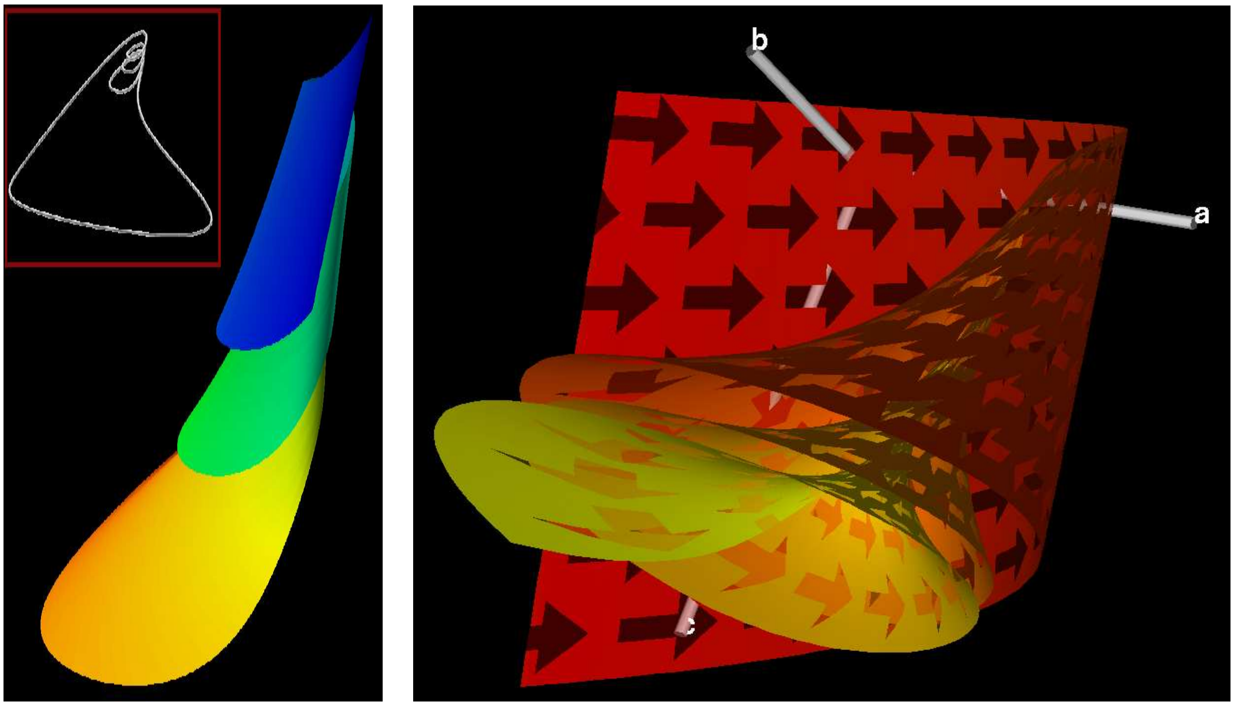
[12] P. Rheingans. Opacity-modulating triangular textures for irregular surfaces. In *Proceedings Visualisation '96*, pages 219–225, 1996.

[13] A. A. Tsonis. *Chaos – From Theory to Applications*. Plenum Press, 1992.

[14] J. J. van Wijk. Spot noise. In *Proceedings SIGGRAPH '91*, pages 309–318, July 1991.

[15] J. J. van Wijk. Implicit stream surfaces. In *Proceedings Visualization '93*, pages 245–252, 1993.

[16] G. Volpe. Streamlines and streamribbons in aerodynamics. In *27th Aerospace Sciences Meeting, Reno, NV*, January 1989. AIAA Paper 89–0140.



(a) (b)

Figure 6: Streamsurface and streamline typical for mixed-mode oscillations (a) – Streamarrows for mixed-mode oscillations (b)

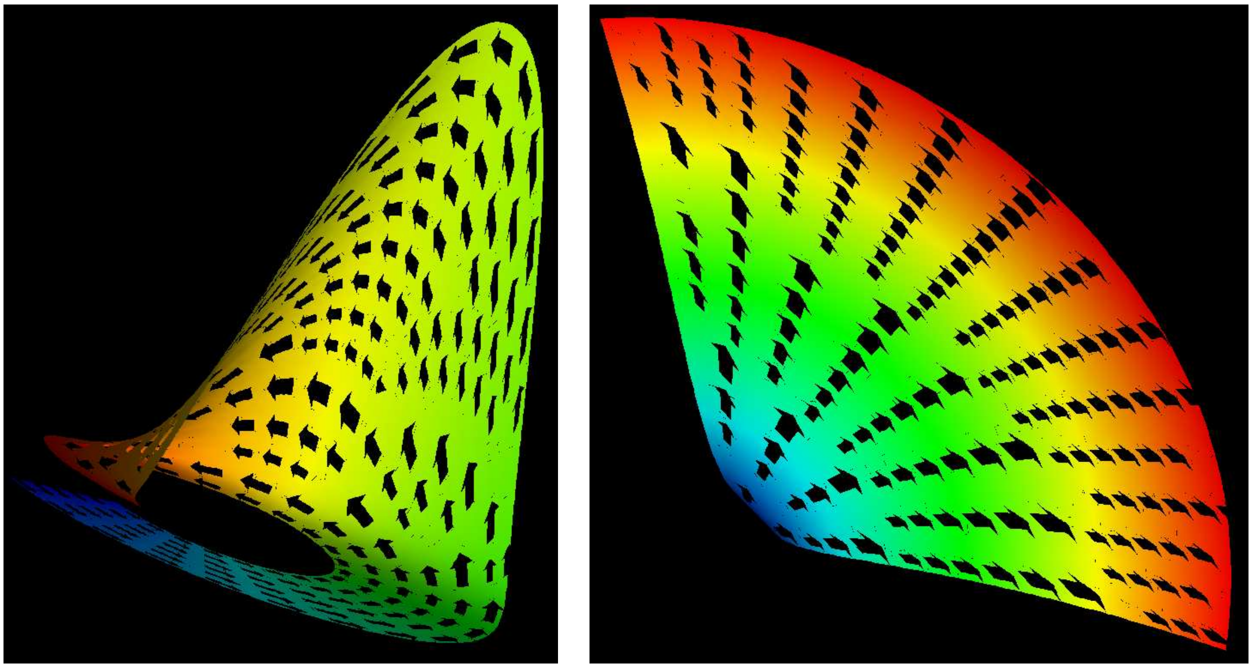
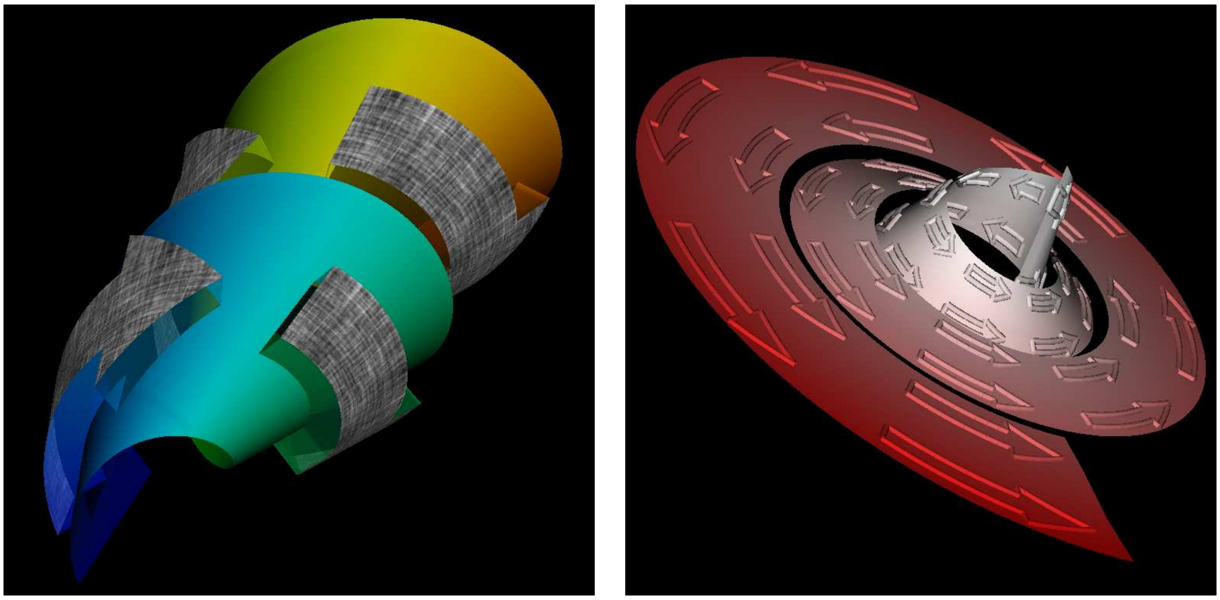


Figure 7: Hierarchical streamarrows, two examples



(a) (b)

Figure 8: Streamarrows shifted out of the streamsurface plus anisotropic spot-noise (a) – Streamarrows represented as 3D tubes plus color coding of velocity (b)

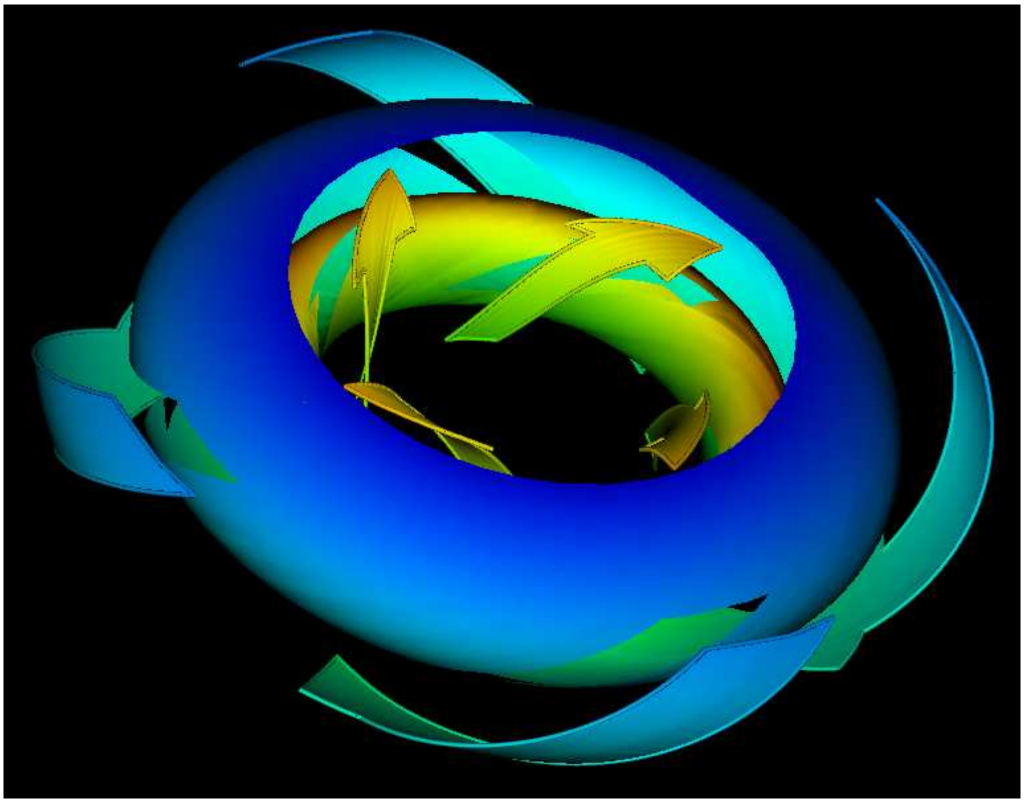


Figure 9: Streamarrows similar to Shaw's hand-drawings