Flow Field Visualization
Using Vector Field Perpendicular Surfaces

Karljohan Lundin Palmerius* Matthew Cooper† Anders Ynnerman‡
Norrköping Visualization and Interaction Studio,
Linköping University, Sweden

Abstract

This paper introduces Vector Field Perpendicular Surfaces as a means to represent vector data with special focus on variations across the vectors in the field. These surfaces are perpendicular to streamlines, with the vector data always being parallel to the normals of the surface. In this way the orientation of the data is conveyed to the viewer, while providing a continuous representation across the vectors of the field. This paper describes the properties of such surfaces including an issue with helicity density in the vector field, an approach to generating them, several stop conditions and special means to handle also fields with non-zero helicity density.


Keywords: vector field visualization, perpendicular surfaces, surface representation

1 Introduction

The search for better ways to visualize complex structures and dynamics in volumetric data is an active area of research. Examples of applications driving this development are the computational fluid dynamics (CFD) analysis of aerodynamics and combustion. There exists a wide range of approaches for visualizing vector fields, see e.g. [Schroeder et al. 1998; Post et al. 2003; Laramee et al. 2004; Laramee et al. 2006]. The popular integration-based techniques — streamlines, stream-ribbons, stream-surfaces, etc. — represent the data at discrete regions in space, but describe the orientation continuously along the direction of the field. A similar effect is achieved using more recent methods, such as dye advection [Weiskopf 2004] or volume LIC [Interrante and Grosch 1997; Interrante and Grosch 1998]. The distribution is discrete across the vectors in the field and the representation continuous along the vectors. Therefore, the streamline and its peers can be thought of as visualizing the data with special focus on variations along the field.

These stream-based approaches to visualizing vector data are generally considered both intuitive and powerful. Sometimes, however, the variation across the vectors in the field, instead of along the vectors, is of more interest. This can be, for example, along the front edge of a wing to visualize variations in wind bifurcation, in flow data to emphasize vortex structures, or in magnetic fields to visualize how a change of position might affect the direction of an attractive force.

This paper introduces Vector Field Perpendicular Surfaces, as a perpendicular analogue to streamlines, as a complement for representing the field data with particular focus on the cross-field variations. When a streamline grows from a seed point forwards and backwards in the orientation of the vector field, this surface grows outwards, perpendicular to the field and such streamlines. In the same way that a streamline can be used to represent vector data and describe the behaviour of the properties along the local vectors, the vector field perpendicular surfaces can be used to represent the same data and describe the behaviour across the orientation of the vectors. A local surface patch representing the orientation of the data is in the work presented here generated by starting at a seed position and marching a surface edge outwards and perpendicular to the vector data.

The contributions of this paper are:

• The introduction of Vector Field Perpendicular Surfaces as the perpendicular analogue to streamlines
• The description of common properties of such surfaces and visualization examples
• Support for the surface representation of vector fields with non-zero helicity density
• Several appropriate stop conditions and suggestions on how to implement them
• A straightforward and fast approach for generating the surfaces
• An open source implementation

2 Related Work

Apart from dense methods, such as LIC [Cabral and Leedom 1993] that display changes in any represented direction, there have been few approaches presented that allow for focus on changes across the field. The probe presented by Leeuw and van Wijk in [de Leeuw and van Wijk 1993] is capable of displaying the local change across and along the field for discrete probed points. This is done by calculating a velocity gradient tensor at the position of the probe and from this extract properties, such as curvature, that are mapped to the visual appearance of the probe. It should be noted that this probe only visualizes the properties at one point, only an analytical extraction of the changes perpendicular to the field and not the changes themselves over larger areas.

Weinkauf and Theisel describe in [Weinkauf and Theisel 2002] Normal Surfaces, surfaces that have the vector field as normals, and
describe many properties of these. They do not, however, extract the surfaces or use them for visualization. As another example of features perpendicular to the vector field, Bachthaler and Weiskopf describe in [Bachthaler and Weiskopf 2007] the use of an LIC that is orthogonal to the flow. This is also combined with animation to increase the perception of features in the visualization.

Zhang et al.[Zhang et al. 2003] presented a method they call “streamsurfaces” (not to be confused with the stream-surfaces mentioned above), where a surface is generated along the two major eigenvectors of a tensor field, starting at a seed position. This results in a visual appearance similar to our method, however their method cannot correctly handle helicity. When visualizing diffusion tensor data this might not produce noticeable artifacts, however the occurrence of helicity must be correctly handled in the generalization of this concept. A continuous surface that is perfectly orthogonal to the vector field can only be defined if the field has zero helicity density in the local region.

Sondershaus and Gumhold[Sondershaus and Gumhold 2003] present an alternative use of the surface representations presented by Zhang et al. In it they distribute points over the surfaces instead of rendering them directly, to avoid occlusion, and apply lighting models and colouring to emphasize shapes and structures. They also present an algorithm for finding such surfaces, using face-based coding. Just like Zhang et al., however, they fail to acknowledge or handle the issue with helicity.

3 The Basic Principles

The vector field perpendicular surfaces method for vector field visualization represents the data visually using surfaces that are, at any point on the surfaces, perpendicular to the vector field. Through the orientation of the surface, they describe the structure of the field in a continuous manner across the vectors in the field. By inspecting the surface across the field, the variations of the field can be visually explored, see example in figure 7. Such surfaces can be generated as patches from discrete seed points the same way that streamlines, for example, are distributed in a flow field.

We can define the perfect vector field perpendicular surface as a continuous set, $\mathcal{S}$, of points, $\vec{x}$, with the following property:

\[
\lim_{|\vec{x} - \vec{x}_0| \rightarrow 0} \frac{\vec{V}(\vec{x}_0) \cdot (\vec{x} - \vec{x}_0)}{|\vec{V}(\vec{x}_0)| |\vec{x} - \vec{x}_0|} = 0 \tag{1}
\]

where $\vec{V}$ is the vector field and where $\vec{x}_0$ and $\vec{x}_1$ are points on the surface. As described in [Weinkauf and Theisel 2002], such surfaces are undefined at critical points, never intersect or touch each other, and also never self-intersect. This simplifies the surface construction since such special cases need not be considered. They are also undefined where the flow is of zero magnitude.

3.1 Non-zero Helicity

Putting the definition of the vector field perpendicular surfaces in equation 1 in a Riemann sum it follows that

\[
\int_{C} \vec{V} \cdot d\vec{s} = 0 \tag{2}
\]

for any curve, $C$, on the surface. A way of writing curl explicitly as rotation is

\[
\vec{n} \cdot \vec{V} \times (\vec{p}) = \lim_{r \rightarrow 0^+} \frac{1}{\pi r^2} \int_{C_r} \vec{V} \cdot d\vec{r} \tag{3}
\]

where $C_r$ is a circle around $\vec{p}$ with radius $r$ and $\vec{n}$ is the normal of that circle. By letting $C_r$ be a curve in the surface defined by equation 1 we see that $\vec{n} = \vec{V}(\vec{x}_0)/|\vec{V}(\vec{x}_0)|$ and thus, using equations 2 and 3, we get

\[
\vec{V} \cdot \vec{V} \times \vec{V} = 0 \tag{4}
\]

which describes a helicity free field. This means that the surface can only be defined if the field has a zero helicity density in the region of the surface. If the vector field is conservative, i.e. $\vec{V} \times \vec{V} = 0$ in which case it will also be helicity free, the surface perpendicular to the vector field will be a section of an iso-surface of the potential of the field, with the iso-value being the potential at the surface.

If there is helicity in the field at the location to be visualized, then a modification of the mathematical definition must be made. Otherwise it is impossible to generate such a visual representation. We define a non-perfect vector field perpendicular surface as a piece-wise continuous, connected set, $\mathcal{S}^\dagger$, of points which has the following property:

\[
\lim_{|\vec{x} - \vec{x}_0| \rightarrow 0} \frac{|\vec{V}(\vec{x}_0) \cdot (\vec{x} - \vec{x}_0)|}{|\vec{V}(\vec{x}_0)| |\vec{x} - \vec{x}_0|} \leq \epsilon \tag{5}
\]

where $\epsilon$ is a small number indicating the maximum level of error and where $\vec{x}_0$ and $\vec{x}_1$ are points in the surface. Since there are no perfect perpendicular surfaces in fields with a non-zero helicity density the visual representation with vector field perpendicular surfaces of such vector fields must allow for a certain orientation error. The maximum orientation error, $\varphi$, for a surface at a certain position can be calculated from equation 5 as $\varphi = \arccos \epsilon$.

By allowing a small error, the surface can describe the flow structure over a small region, however as the surface grows the amount of error will increase. To ensure a bound error in the surface orientation we must tear the surface into smaller pieces, each of which provides a good representation of the flow. The size of these pieces will be dependent on the amount of curl, that is how close to zero the vector field helicity density is in the local region, and the amount of error allowed in the visual representation.

3.2 Standard Patterns

The surfaces react in a predictable way to form identifiable patterns for certain structures in the vector data, see figure 1. Divergence and convergence in the flow are rendered as cone or spherical cap patterns, depending on the radial structure of the divergence or convergence, see figure 1(a). The combined divergence in one direction and convergence in the other also produces a very distinct pattern, see figure 1(b).

These three patterns, convergence, divergence and combined, are similar to what you might get with Leeuw and van Wijks flow field probe[de Leeuw and van Wijk 1993] if the data at the probed point are similar to the surrounding points described by the larger surface. A rotational field, however, typically has a non-zero helicity and therefore requires tearing of the surfaces which produces a distinct fan-like pattern, see figure 1(c). Each patch in the teared surface describes the local flow direction and the combined picture shows the rotational nature of the flow. Also, the amount of tearing indicates the magnitude of curl in the area.

Observe that these are only typical patterns and that a single vector field perpendicular surface can show several of these patterns describing the variations across the vector field.
3.3 Vector Property Extraction

The vector perpendicular surfaces can also be used to visualize other properties of 2D manifold nature extracted from scalar, vector or tensor data. Zhang et al. have already shown in [Zhang et al. 2003] that creating a surface perpendicular to the minor eigenvector of a diffusion tensor field can provide an intuitive visualization of the planar class tensors in the data. Other properties represented by vectors can be extracted and visualized in a similar manner. For example, the gradient of scalar data can be used to control the surface orientation, which results in iso-surface patches around the selected seed points.

In flow data an example of a property of interest is the location and shapes of helices. This property can be estimated by calculating the cross product between the flow and the vorticity, as shown by Lawrence et al. in [Lawrence et al. 2000].

$$ \vec{V}_h(x) = \left( \vec{\nabla} \times \vec{V}(x) \right) \times \vec{V}(x) $$

where $\vec{V}$ is the vector data. Using this property, $\vec{V}_h$, to define the surface orientation, we get the examples shown in figure 2.

4 Fast Surface Construction

Vector field perpendicular surfaces are generated, just like streamlines, from seed points distributed in the data. Just like with streamlines these seed points may be pre-defined or automatically or interactively distributed. From each such seed the surface is computed by first generating an initial seed surface. The sequence of rim points of this surface defines an edge, $E$. The surface is then iteratively grown by producing new points defining a new edge outside the previous one, see figure 3(a). We call each such iteration a ring, starting with ring zero being the initial, closed edge around the seed point.

Since points are iteratively added in a structured manner it is straightforward to define triangles that connect these points. The triangles then become well structured and it is simple to store the surface in the form of triangle strips, a structure that is more memory efficient and faster to render than unordered triangles.

A variety of stop criteria are defined for the surface, for example limiting the length of growth and thereby the size of the surface, see section 5 for details. When a stop criterion is fulfilled for a certain point at the edge, the surface should be terminated at that point, but may continue the propagation on both sides of the point, see figure 3(b). To handle this the edge is divided into segments, $S$. Thus, conceptually the edge at ring $R$, $E_R$, consists of one or several edge segments, $E_R = \{S_0^R, \ldots \}$. Each segment, in turn, consists of at least three points, $S_0^R = \{p_0, p_1, p_2, \ldots \}$, a start point, one or more middle points and an end point. Why each segment consists of at least three points will be apparent from the approach chosen to define new surface points.
4.2 Edge Propagation

For each segment in the edge of a ring, \( R \), new segment(s) are generated defining a new edge of the subsequent ring, \( R + 1 \). Thus, the edge propagates through the data and the surface grows ring by ring. To generate new points for a segment in the subsequent ring the points in each segment in the current ring are processed in turn, see figure 3(a). When a point is processed the ‘triplet’ made up of the preceding, current and subsequent points is used. Since a segment contains start, end and middle points, we are guaranteed to have at least one triplet to process. To determine the number of points to generate for the subsequent ring we first estimate the available space in front of the triplet of points, \( \vec{p}_{j-1, j, j+1} \).

\[
\alpha = \begin{cases} 
\arccos \frac{\hat{q}_t \cdot \hat{q}_j}{\| \vec{p}_{j-1} - \vec{p}_j \|} & \text{if } \hat{q}_t \times \vec{V}(\vec{p}_j) \geq 0 \\
2\pi - \arccos \frac{\hat{q}_t \cdot \hat{q}_j}{\| \vec{p}_{j+1} - \vec{p}_j \|} & \text{if } \hat{q}_t \times \vec{V}(\vec{p}_j) < 0
\end{cases}
\]  

Points are added by filling the available space in front of each triplet. The point separation should, as before, be as close to 60 degrees as possible, see figure 5, so the number of points to generate, \( N_{\text{new}} \), and the point separation, \( \beta \), are determined by

\[
N_{\text{new}} = \left\lceil \frac{2\alpha}{\pi} \right\rceil \quad \beta = \frac{\alpha}{N_{\text{new}} + 1}
\]

where the angle bracket signifies rounding to the closest integer value. The first point position is then determined by rotating \( \hat{q}_t \) by \( \beta \) around the local vector and performing one step, as described in section 4.1, in that direction. The positions of any further points are determined the same way by again rotating the vector \( \hat{q}_t \) by \( \beta \) and then performing one step from the centre point, \( \vec{p}_j \), of the triplet.

With this fundamental approach the last point generated from one triplet will almost coincide with the first point of the subsequent triplet, see figure 5. The mean of these two points is used as the point connecting the two triplets. This approach allows us to use a flexible approach to the estimate of the surface growth length (section 5.1). Observe that one reason that these two points do not always coincide is that in fields with non-zero helicity the integration over two different paths towards the same position will not intersect, as described in section 3.1. Even though these two paths are both orthogonal to the vector field the surface connecting the two paths will not be, which introduces a small error in the surface orientation compared to a perfectly perpendicular surface.

Curvature in the data can cause the base of the triangles in a region to become shorter and shorter for each ring, see figure 6. To avoid the case where this may cause a triangle to become obtuse (having a point outside the base) the following condition is added:

\[
\hat{q}_t \cdot (\vec{p}_{\text{new}} - \vec{p}_j) < |\vec{p}_{j+1} - \vec{p}_j|
\]
Only if this condition is true will the point fall within the base of all triangles associated with the current point, \( \vec{p}_j \). By not using points for which it is false the points distribution can be kept structured, see figure 6.

When all segments for an edge have been processed and an edge for the subsequent ring has been generated, it is then necessary to close the ring of that new edge. Otherwise, the resulting ring will have a notch, a cleft in the edge, where the first and last segment share their start and end point, see figure 3(b). If this shared point fulfills a stop condition, the ring is not closed and a flag is set so that it will not be closed in subsequent rings either. If the point is a valid candidate for propagation, however, new points are first added to fill the cleft if necessary, as described above. The last point added is then used as the new start and end point for the first and last segment. After this the edge is fully connected at the start/end point, similar to the initial hexagon surface in figure 4.

### 4.3 Summary

The steps described above can be summaries in pseudo code in the following way. The same notation is used here as is used above.

```plaintext
1. initialize \( E_0 \) and set \( R = 0 \)
2. while \( E_R \) is not empty
   3. for each \( S_i \) in \( E_R \)
      4. add \( \vec{p}_0 \) to new segment \( S_k^{R+1} \)
      5. for each triplet \( (\vec{p}_{j-1}, \vec{p}_j, \vec{p}_{j+1}) \) in \( S_i \)
         6. if stop criterion met for \( \vec{p}_j \)
            7. add \( \vec{p}_j \) to \( S_k^{R+1} \)
         8. if size(\( S_k^{R+1} \)) > 2
            9. add \( S_k^{R+1} \) to \( E_{R+1} \)
         10. increment \( k \)
         11. add \( \vec{p}_j \) to \( S_k^{R+1} \)
         12. continue from line 5
      13. estimate angle \( \alpha \) between \( \vec{q}_l \) and \( \vec{q}_r \)
      14. \( N_{\text{add}} = \left\lceil \frac{\pi}{\alpha} \right\rceil \)
      15. do \( N_{\text{add}} \) times
         16. rotate \( \vec{q}_l \) by \( \beta \)
         17. \( \vec{p}_{\text{new}} = \vec{p}_j + \vec{Q}(\vec{x}, \vec{q}_r) \)
         18. if first of the \( N_{\text{add}} \) points and len(\( S_k^{R+1} \)) > 1
            19. merge \( \vec{p}_{\text{new}} \) with last added point
         20. else if \( \vec{q}_r \) \((\vec{p}_{\text{new}} - \vec{p}_j) < |\vec{p}_{j+1} - \vec{p}_j| \)
            21. add \( \vec{p}_{\text{new}} \) to \( S_k^{R+1} \)
         22. add \( \vec{p}_{j+1} \) to new segment \( S_k^{R+1} \)
         23. add \( S_k^{R+1} \) to \( E_{R+1} \)
      24. if is_closed
         25. stop criterion met for \( \vec{p}_0 \)
         26. is_closed = false
         27. else
            28. join \( S_0^{R+1} \) and \( S_k^{R+1} \)
      29. increment \( R \)
```

### 5 Stop Criteria

The vector field perpendicular surfaces are generated by starting at a seed and propagating outwards. The propagation of the surface stops at a point on the rim when a stop criterion is fulfilled at that point. An obvious such stop criterion is the lack of data: if the local vector is shorter than some threshold value, the uncertainty is deemed too high and the surface is terminated at that point.

This section describes additional stop criteria and how they are estimated at run-time. Observe that the stop criteria introduce parameters that, just like the step length, can be adjusted to fit the data and the purpose and properties of the visualization.

Terminating at a point leaves a cleft of approximately 60 degrees in the edge, see figure 3(b). If the stop criterion is not fulfilled for the points around that cleft, the surface may start to grow inwards through the cleft, resulting in a self intersecting surface growing simultaneously in all directions around the point of surface termination. This can be avoided by simply not expanding the points next to a terminated point. By doing this not directly after the termination, but after one or a few rings, the level of sparseness of the resulting surface can be somewhat controlled. This measure has been taken in the implementation used to make the example images.

#### 5.1 Length Limit

The length limit is a useful stop criterion to avoid creating a surface that grows to infinity. Therefore each point, \( \vec{p}_j \), is associated with an accumulated length, \( L_j \). Each time a new point is generated its accumulated length is estimated as the accumulated length of the point being expanded (described in section 4.2) plus the length of the step. Since the steps made follow a triangle pattern and are generally not performed in the direction of the surface expansion (see figure 5) the length increment is subject to a modulation:

\[
L_j = L_i + \delta_{\text{step}} \cos \left( \beta - \frac{\alpha}{2} \right) \tag{14}
\]

where \( L_i \) and \( L_j \) are the lengths associated with the old and new point, respectively. When two points are merged the edge growth direction of these two points will be different and thus the estimated length will differ even though the points almost coincide. The mean length of the two points is then used as an estimate for the new point.

When the accumulated length, \( L_j \), exceeds a preset length threshold, \( C_l \), the surface should be terminated at the associated point. To get a smooth edge of the final rim, however, the step length is adjusted at the last step so that the last rim points end up at the exact
length specified:

\[ \delta_{\text{step}} = \delta_{\text{step}} \frac{C_L - L_j}{\cos (\beta - \frac{\pi}{2})} \] (15)

In special cases, where a triplet forms a very sharp angle, the cosine term (equation 14) may cause a length decrement instead of an increment. To avoid this a lower limit may be set on that term.

5.2 Maximum Winding Angle

A vector field perpendicular surface can in a high curvature field describe a closed loop which can be visually confusing. The winding angle, \( \omega_j \), associated with point \( \vec{p}_j \), is used to add control over this property of the surface. It is estimated by comparing the direction of propagation to and from the point \( \vec{p}_j \). For each point the accumulated winding, \( \Omega_j \), is tracked so that the surface can be terminated at the point if \( \Omega_j > C_\Omega \), where \( C_\Omega \) is a preset accumulated winding limit.

Too large a change in vector orientation in a field may introduce too large an error in the integration, or may simply cause strange surfaces. A winding limit, \( C_\omega \), is also introduced that makes it possible to terminate the surface in regions containing too large a change.

5.3 Maximum Orientation Error

As is described in section 3.1, only vector fields with zero helicity density will have surface manifolds that are at all points perpendicular to the field. Fields with regions of non-zero helicity will need a certain allowed orientation error for a continuous surface to be generated. In certain cases the error might become too large for an appropriate representation of the data, so an enforced limit, \( C_\phi \), of the orientation error, \( \phi \), must be introduced.

We estimate the orientation error associated with a point, \( \vec{p}_j \), as the deviation from a right angle of the angle between the normalized local vector and the direction to the subsequent point in the segment

\[ \phi = \arcsin \left( \frac{\vec{V}(\vec{p}_j) \cdot (\vec{p}_j - \vec{p}_{j-1})}{|\vec{V}(\vec{p}_j)| |\vec{p}_j - \vec{p}_{j-1}|} \right) \] (16)

When this value exceeds \( C_\phi \) the surface is terminated at that point. Since the current edge segment becomes divided into two segments around this point, the surface will continue to propagate on both sides. This will produce an effect similar to tearing the surface to make it fit the orientation of the vector field. This effect is clearly visible in figure 7.

6 Results

The implementation of the presented work has been used to produce visualization examples for demonstration purposes and to evaluate the algorithm performance.

The vector field perpendicular surfaces approach to vector visualization has been implemented in C++ using H3D API, an X3D-based scene graph system for implementing multi-modal interaction, and the Volume Haptics Toolkit (VHTK) which is a H3D API toolkit for volume haptics and visualization. This reference implementation is freely available in the source distribution of the GNU GPL release of VHTK.

The surface generating scene graph node extends the IndexedTriangleSet type, which handles point coordinates, point colours, and an index array that specifies the point structure for building triangles. The surface colour is controlled through an RGB transfer function from the magnitude of the vector field. It is, however, possible to replace this with any scalar property, or even another scalar field.

The integration of the function in equation 7 is in this implementation estimated using fourth order Runge-Kutta. Each step onwards in the algorithm is performed using one single integration step. Thus, the step length, \( \delta_{\text{step}} \), is here both describing the triangle size in the final mesh and the step length in the Runge-Kutta integrator.

6.1 Examples

The first data set used for the demonstration comes from a CFD simulation of the air flow around an experimental unmanned aerial vehicle (UAV) called SHARC. The vector field used to represent the air flow is 32 bit floating point vectors and has a resolution of 128³ voxels. An example of the visualization of this data set using vector field perpendicular surfaces is shown in figure 7.

Experience with the vector field perpendicular surfaces in real data sets indicates that it is possible to learn how to recognize these patterns and associate them with structures in the vector data for improved understanding of, for example, complex flow. The air flow data is almost free from helicity in some regions in which the surface plainly forms after the behaviour of the flow. This can be, for example, the convergence in the flow indicated by the surfaces in the upper circle in the figure. In other regions, especially near vortices, the vector field has helicity and the surface is torn to fit the data, as shown in the lower circle. A maximum orientation error of 5 degrees has been allowed, which is close to the perceptual limit[Knill and Kersten 2003].

For a second example the blunt fin CFD data set[Hung and Buning 1984] has been resampled to a regular grid of 256³ voxels with 32 bit floating point vectors in order to fit the currently used framework. The visualization of this data set, see figure 8, shows how the cross sections represented by the vector field perpendicular surfaces visualize the change in flow direction at different levels from the surface of the fin. Stream-tubes rendering has been added to provide a comparison with classical vector visualization. As can be seen the surfaces display, through their orientation, how the flow direction varies across the flow, while the stream-tubes show how the flow turns along the flow.

6.2 Performance

The time it takes to generate a vector field perpendicular surface is clearly dependent on the step length, \( \delta_{\text{step}} \), and the allowed length of propagation, \( C_L \), since these have direct impact on the number of points that need to be generated. Also, the amount of tearing in the surface and the maximum winding allowed have an impact. The performance of the algorithm, when generating the image in figure 7, is presented in figure 9 as a graph displaying the time it takes to generate each surface when using different length limits. The step length is 0.005 m and the surface length limit for the example images is 0.1 m. Observe that since the number of points to generate on a surface relates to the square of the surface radius, the delay is close to linear in the log-log scale.

Because the surface is a generated 2D structure it will always be slower than generating a streamline, which is only a 1D structure.
Figure 7: This visualization of the SHARC CFD simulation shows both the convergence pattern (upper circle) and the vortex pattern (lower circle), even though the streamlines used to seed the vector field perpendicular surfaces do not directly cross the centre of the vortex. The orientation of the surface at any position shows the orientation of the flow, both around the vortices and at other locations in the flow.

Figure 8: This visualization of the Blunt Fin CFD data set shows how the change in flow direction at different levels displays in a twisted surface and that the curl causes the surface to tear. The stream-tubes rendering provides a comparison with classical vector visualization.

Figure 9: The time required to generate a single vector field perpendicular surface compared to the length limit of the surface, when the step length is set to 0.005 m.

Even generating stream-tubes or stream-surfaces should have a time complexity of $O(L)$, where $L$ is the length limit. For the vector field perpendicular surfaces the length limit specifies the radius, so generating such a surface will typically be done with a time complexity of $O(L^2)$.

7 Conclusions

While there exist a wide range of methods for vector field visualization, few of them produces a generally applicable representation of the data with special focus on the changes across the orientation of the vectors in the field. The vector field perpendicular surfaces introduced in this paper complement this array of methods by rendering a surface across the vectors, with surface orientation perpendicular to the field, a perpendicular analogue to streamlines. The surface is generated in a structured manner, resulting in a straightforward triangulation, little need for caching and no search for neighbouring points. We have also shown that the implemen-
tation provides stable rendering and is fast enough to be used for interactive visual data exploration. The issue with errors in surface orientation inherent for regions with helicity in the vector field has been partly solved using a stop criterion that results in a clear tear in the surface.

The examples and identified standard patterns show promising results and potential use for certain situations, especially where the search for special properties is most effectively performed orthogonal to the vectors of the data. The visual representation of the data is, perhaps, not as intuitive as streamlines, but is straightforward and provides a direct connection between the properties in the field and the visual appearance. It should be noted, however, that to be able to generate a perpendicular surface representation of vector data with non-zero helicity density requires some orientation error. The amount of error that can be allowed without introducing a negative impact on the interpretation of the data is suspected to be much higher than the perceptual limit and is a topic for future research.

References


Schroeder, W., Martin, K., and Lorensen, B. 1998. The Visualization Toolkit, 2 ed. Prentice Hall PTR.


©ACM 2009. This is the authors version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in Proceedings of Spring Conference on Computer Graphics, 2009