Straightening Tubular Flow for Side-by-Side Visualization

Paolo Angelelli, *Student Member, IEEE*, and Helwig Hauser, *Member, IEEE*

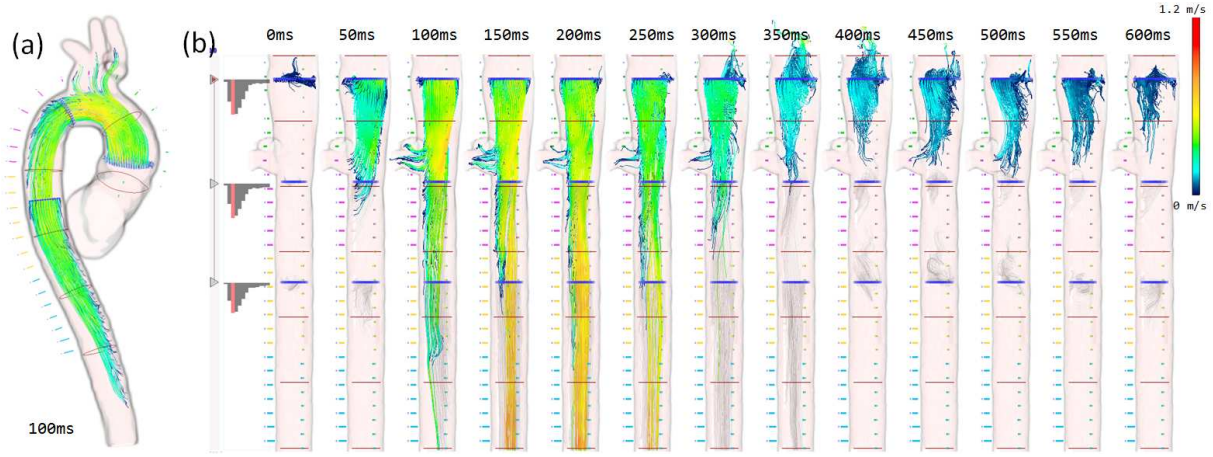

Fig. 1. (a) A timestep of an aortic flow dataset in its anatomical context, rendered using a conventional streamline visualization. (b) Side-by-side visualization of the straightened vector field, showing all the timesteps juxtaposed. The streamlines traced from the first seeding plane are rendered in focus, and the others in grey as context.

Abstract—Flows through tubular structures are common in many fields, including blood flow in medicine and tubular fluid flows in engineering. The analysis of such flows is often done with a strong reference to the main flow direction along the tubular boundary. In this paper we present an approach for straightening the visualization of tubular flow. By aligning the main reference direction of the flow, i.e., the center line of the bounding tubular structure, with one axis of the screen, we are able to natively juxtapose (1.) different visualizations of the same flow, either utilizing different flow visualization techniques, or by varying parameters of a chosen approach such as the choice of seeding locations for integration-based flow visualization, (2.) the different time steps of a time-dependent flow, (3.) different projections around the center line , and (4.) quantitative flow visualizations in immediate spatial relation to the more qualitative classical flow visualization. We describe how to utilize this approach for an informative interactive visual analysis. We demonstrate the potential of our approach by visualizing two datasets from different fields: an arterial blood flow measurement and a tubular gas flow simulation from the automotive industry.

Index Terms—Flow Visualization, Data Reformation, Comparative Visualization.

1 Introduction

Tubular flows are studied in many fields, such as in medicine and engineering. The visual exploration and analysis of such flow data can be challenging, due to the often varied geometry and topology of the flow, and due to a larger number of aspects of the data that are of interest, in particular in time-dependent flow. These aspects include various scalar attributes, such as flow velocity, pressure and vorticity (see Section 2 for a collection of surveys on this topic), as well as derived attributes. On the visualization side, the variation of seeding structures, integration length and the type of primitive for an integration-based visualization, different time steps of the flow, and the variation of other visualization parameters are also aspects of interest.

To enable an analysis that is based on several such aspects, it becomes interesting to consider different views on the data as well as the relation between these views. Different strategies for integrating different visualizations have been proposed: interactive tools for the visual exploration with multiple, coordinated views [6, 20], the fusion of different visualizations in the same view [8, 11, 22], and placing different views side-by-side [32]. Image fusion techniques are powerful tools, as they can visualize multiple aspects of the data in the same reference frame, thus allowing to easily and effectively relate them to each other. On the downside, there are rather limiting restrictions on how much can be fused in a single image. Side-by-side visualizations, instead, can integrate more views of the data, only limited by the overall available space. Moreover, being thess visualization simpler when compared to others, they are generally easier to read and interpret. Additionally, they can also be used to show the same attribute over multiple time-steps or visualized with different parameters, thus enabling alternative types of visual comparison. Last, they can also be combined with image-fusion techniques, leading to side-by-side visualizations of fused views. In terms of limitations, it takes additional space to juxtapose views, so the number of views that can be placed side-by-side is also limited. Second, relating separated views is not a straightforward process, as they are not specified in a common reference frame anymore. Previous work [32] suggests that the question of whether or not to use side-by-side visualization also depends on the application at hand and on which advantages/disadvantages to prioritize. Our contribution addresses the cases where a side-by-side visualization is preferred.

We propose a new solution to the side-by-side visualization of tubular flow datasets. In order to effectively juxtapose views of tubular flows, we introduce the concept of straightening the flow visualization

- *Paolo Angelelli and Helwig Hauser are with the department of Informatics at the University of Bergen, E-mail: {paolo.angelelli, helwig.hauser}@uib.no.*

Manuscript received 31 March 2011; accepted 1 August 2011; posted online 23 October 2011; mailed on 14 October 2011.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

(e.g., streamlines or pathlines) along the center line of the bounding tubular structure, often being the main reference direction of the flow. Using this approach multiple views can be aligned with one axis of the visualization and made parallel to each other along the straightened center line. With such a side-by-side layout it becomes possible to relate different views in the visualization in a straightforward way, as well as making the visualization more compact, allowing to have more views at the same time.

In this paper we first describe how to realize such a straightened visualization. Then, we show how the presented approach has been used, in Section 4, to visualize two tubular flow datasets: a Phase-Contrast Magnetic Resonance Imaging (PC-MRI) scan of a human aorta, containing time-dependent measurements of the blood flow, and a CFD simulation of the exhaust system of a racing engine. Conclusions and future work are presented in Section 5. A discussion of related work is presented in the next Section.

2 Related Work

Flow visualization is an active research topic for over two decades. An extensive body of related literature exists, and many useful surveys exist as well. Post et al. [25, 26], as well as Laramee et al. [13, 14], Peng and Laramee [23], McLoughlin et al. [19], Salzbrunn et al. [30] and Pobitzer et al. [24] have published extensive and informative surveys on different aspects of flow visualization.

Considering specifically the visualization of tubular flow, Nobrega et al. [21] simulated tubular flow in its context, relying on the centerline of the boundary structure for which they propose a novel extraction algorithm. Lež et al. [17] propose an interactive visual analysis approach for studying pathlines, using projections of the dataset for the selection process and to cope with the complex topology of the flow and its tubular context. More domain-specific work has been done, in particular in the field of medical visualization. Van Pelt et al. [31] incorporated illustrative visualization techniques in an application for visualizing blood flow in the aorta and other large vessels, introducing flow-rate arrow trails. Markl et al. [18] presented a comprehensive 4D visualization of the blood flow in the heart and great vessels by using glyphs, streamlines and pathlines, as well as exploded views with information visualization techniques.

One of our main goals was to enable an efficient comparison of the different aspects of the tubular flow data. Previously, Verma and Pang [32] presented a tool for comparing flow data. An important contribution of their work is the distinction of three possible levels of comparison in flow visualization: image-level, data-level and feature-level. They describe the major drawback of image-level comparison as "it leaves the burden on the users to identify regions of difference and to quantify the differences themselves". Our approach eases the comparison by using the main direction of the flow to align multiple visualizations, paralleling each other, so that it becomes straightforward to relate the side-by-side views. Jones and Ma [9] have also adopted a similar concept to ease image-level comparison, by projecting integrated lines onto the three Cartesian planes.

Relevant work has also been done for reforming tubular structures into a plane, also here in particular in the field of medical visualization. Vilanova et al. [34] perform a 2D reformation of 3D human colon data. They extract the colon centerline, and use it for performing nonlinear radial raycasting, producing a flattened view of the internal wall of the colon. Kanitsar et al. [10] presented curved-planar reformation (CPR) approaches for entire vascular trees. Borkin et al. [2] also created projections of the coronary artery tree, mapping it to a 2D tree chart, where each vessel is straightened and depicts its endothelial shear stress. Ropinski et al. [27] applied flattening techniques to volumetric scans of mice aortas, to provide a navigational tool that links 2D and 3D visualizations of their multimodal dataset. Curved-planar reformations has also been applied to other anatomical organs. Vrtovec et al. [35] applied CPR to human spine datasets: this work enabled the comparison of all the vertebrae in a single visualization, without the need of slicing through the volumetric scan. Daae Lampe et al. [12] presented a new technique to perform curve-centric volume reformation (CCVR), straightening the original 3D scalar data into a new volume, centered around a 3D curve. This can be considered as warping the space, and previously Chen et al. [3], as well as Correa et al. [5], proposed generalized space warping methods, based on spatial transfer functions and generalized displacement mapping.

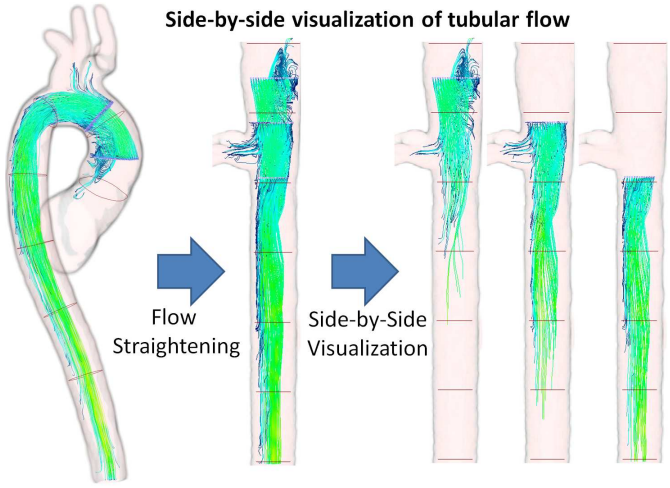

Fig. 2. Illustrative overview of the proposed approach to realize a side-by-side visualization of tubular flow based on straightening the flow domain. In the side-by-side visualization the seeding structure has been varied in order to study different seeding locations.

The method presented in our paper pursues the same purpose, however targeted not only to scalar data, but especially to vector field data. In the following we first describe how to realize straightened side-by-side visualizations of tubular flows before we then demonstrate our approach in the context of two application examples.

3 Method

In the following we present our method for creating straightened side-by-side visualizations of tubular flows, as illustrated in Figure 2. The method can be used to complement regular visualizations of tubular flows, in order to statically visualize multiple aspects of the data at once, including the time dependency. We describe the method in two parts: first, two techniques for generating straightened visualizations of tubular flows, defined as vector fields on a Cartesian grid; then, a set of techniques to assemble these straightened views in order to create efficient side-by-side visualizations. The first part is described in the next Section, while the second part is described in Section 3.2.

3.1 Centerline-centric tubular flow straightening

Conceptually, visualizations of straightened tubular flow can be generated using two different approaches (see Figure 3):

Straightening the flow domain: this approach performs a curve-centric vector field reformation (CCVFR), to generate a deformed vector field, straightened along the centerline of the tubular structure. In the second step, any flow visualization technique can be used directly to visualize this reformed vector field, producing straightened views of the flow. To perform the CCVFR, we introduce a method that extends the algorithm proposed by Daae Lampe et al. [12], such that it can be used to reform vector fields. This method is described in Section 3.1.4.

Straightening the flow visualization: this approach generates the primitives used for visualizing the flow, such as streamlines, pathlines, or more complex visualization cues, in the original flow domain. These generated visualization elements are subsequently deformed into the straightened domain using the centerline as reference. We describe an algorithm to performs this operation on line primitives in Section 3.1.3.

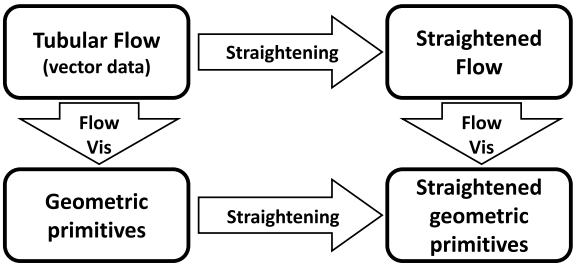

Fig. 3. Two paths to realize a straightened visualization of tubular flow data: straightening flow visualization or visualizing straightened flow.

The advantage of the first approach is the simplicity of producing flow visualizations: once the vector field is reformed, any existing flow visualization technique can be used without modification. This also avoids performance penalties when compared to visualizing the original data. On the downside, the vector field reformation process may introduce numerical inaccuracies. Reforming visualization cues, on the other side, produces an exact straightening of the visualization primitives, at the cost of a higher computational complexity. Moreover, this approach requires a tailored algorithm for each flow visualization technique to be realized. In order to combine the best of both approaches and to avoid the mentioned drawbacks, we realize a hybrid scheme, that renders the straightened vector field data during user interaction, to keep the system interactive. The second approach is then used on demand, to produce an as accurate as possible straightened visualization. Performance and error analysis are described in section 4.3.

3.1.1 Prerequisites

The straightening operation, that is integral to both of the approaches, grounds on the definition of a curvilinear coordinate system that is constructed along and around the centerline through the flow tube. Conceptually, we can consider a *moving frame*, similar to the Frenet frame of a curve [7], following the centerline of the structure bounding the flow, and thereby tracing the curved, centerline-centric, frame of reference for this tubular object. This moving frame is used to extract oriented cross-planes orthogonal to the centerline, and eventually to define a new grid for the data. Details on how to generate this grid within the curved structure are given in Section 3.1.3. Before, however, we describe how to extract the centerline itself, and how to compute the frame along it.

There exist several techniques for extracting centerlines, both from geometric data [21] and from volumetric data [4, 15, 33]. To demonstrate our method in Section 4, we use the approach proposed by Cornea et al. [4], previously also used in other works [27]. This approach operates on volumetric data, and extracts the skeleton of an object using a potential field. The skeleton consists of a set of segments, which need to be connected in order to create the final centerline. For the cases shown in this paper, we extracted the lumen of the tubular objects automatically, by thresholding a scalar volume containing the maximum magnitude of the vectors over all the time steps. For the aorta dataset, this extracted structure has been semi-automatically refined using the ITK-SNAP tool, to increase the accuracy and remove other vessels. However, different automatic techniques for 3D vessel lumen segmentation could also have been used, and Lesage et al. [16] provide a comprehensive survey on the topic. Once we extracted the object skeleton, we computed the final centerline using a tool based on the Visualization Toolkit (VTK), helping to pick and connect together the skeleton segments. This process could also be automatized [27], but, for our purpose, it did not require further refinements.

Given a curve, such as the above-mentioned centerline, several methods for computing moving frames are available, and Daae Lampe et al. [12] provide a useful survey on this topic. In their paper, they also propose a modified version of the Frenet-Serret formulas for computing a moving frame [7], achieving a curve-centric (scalar) volume reformation (CCVR). The Frenet frame is, in fact, limited to twice

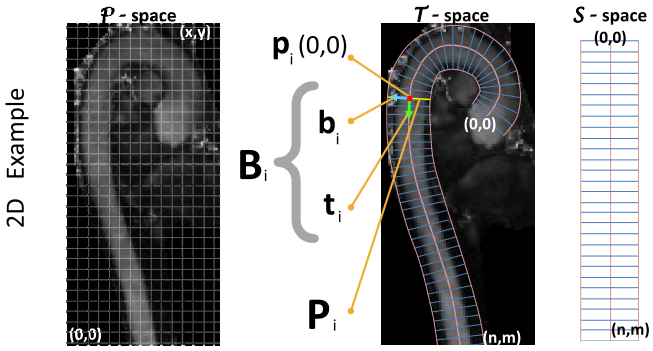

Fig. 4. We consider three spaces: the data is given as a Cartesian grid (space $\mathcal{P}$). A curvilinear grid is constructed along and around the centerline (space $\mathcal{T}$), and after the reformation, this grid becomes a new Cartesian grid (in space $\mathcal{S}$).

continuously differentiable curves. By using a constant, user specified *up vector* to compute the binormal in a curve point, the authors both achieve a fixed frame orientation for the whole reformation, and a definition of the binormal (and subsequently of the normal) also where the curve is straight, and the derivatives would be vanishing. They also convolve the tangent and the normal with a smoothing kernel to prevent an exceeding roughness of the curve. Interpolation, in their case, is performed in spherical coordinates, to prevent abrupt sign changes of the vectors. This technique has, however, the obvious limitation that it is not applicable in those points of the centerline where the normal is parallel to the user-specified up vector.

To overcome this limitation, we extend this method by using a user-specified up vector (that also defines a fixed frame orientation around the centerline) only in the initial point of the centerline. We observed that the centerline is subdivided in segments by a number of evenly spaced positions along the line, depending on the desired amount of orthogonal cross-planes. For the binormal computation in the current position, our method uses the normal in the previous position as the "suggested" up vector. With a smoothly varying tangent and a sufficient density of points, this approach does not incur the case when the normal is parallel to the tangent. Therefore it becomes possible to reform tubular structures without being limited to bends of less than 90 degrees along the normal direction. In our visualizations, we always visualize the flow in its context, e.g., the boundary surface of the tubular structure, that we consider as the primary orientation cue. Therefore, this enhanced computation of the moving frame is also used in our prototype to implement a standard CCVR for the flow context. The CCVR method makes use of quads of user specified side length to bound the resampling of the original data on evenly spaced planes along and around the centerline (not at the least to prevent the resampling in regions where these planes intersect).

3.1.2 Physical space, tubular space and the straightened space

The centerline with its orthogonal cross-planes can be seen as a skeleton bounding the tubular flow. The main idea behind this work consists of using this skeleton to "superimpose" a new curvilinear grid on the data. This grid is used to perform a curve-centric reformation of the vector field, by transforming the vectors with the inverted Jacobian matrix of the grid in each sampling position [29]. This grid is also used to compute "reference" intersection points between integrated lines, such as streamlines, and the cross-planes, in order to map each line to the straightened space.

Let us first formally introduce the three different spaces we are considering. The first space, $\mathcal{P}$, is the original physical space, in which

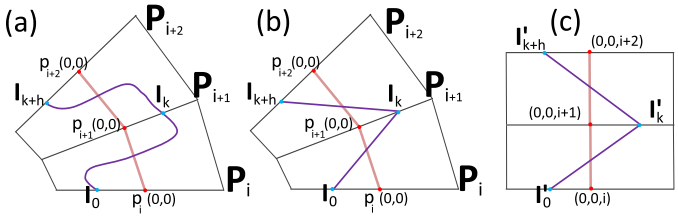

Fig. 5. Intersections (in red) of the line traced from the point $\mathbf{l}_0$ (a) skeleton of the line using only the intersection points (b). Straightened skeleton in $\mathcal{S}$ space, by performing the mapping from $\mathcal{P}$ to $\mathcal{S}$ of the intersections (c).

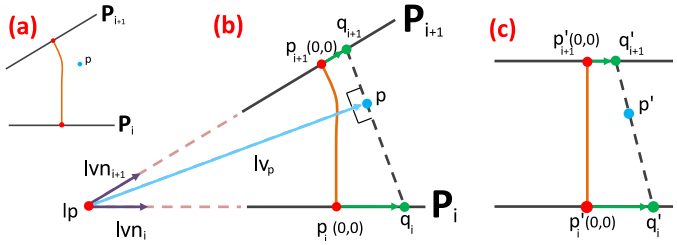

Fig. 6. Example in **2D** of a point $\mathbf{p}$ to be reformed, lying in $sector_i$ (a). Elements of the reformation, needed to compute $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$ (b). Reformed $\mathbf{p}'$ in $\mathcal{S}$ space, computed through the points $\mathbf{q}'_i$ and $\mathbf{q}'_{i+1}$.

the data, the centerline and the modified Frenet frame are defined. The second space, $\mathcal{T}$, is the tubular space defined by the moving frame. Finally, the straightened space, $\mathcal{S}$, is the space produced by the reformation, and it is a Cartesian grid.

Normally, $\mathcal{P}$ is defined by the application, and in the cases presented here it is a Cartesian grid. $\mathcal{T}$, instead, is "traced" by the moving frame along the centerline, which generates a curvilinear grid. Assuming a subdivision of the centerline in n segments of equal length (the number of segments is controlled by the user), there are $n+1$ evenly spaced points $\mathbf{p}_i^{\mathcal{P}}(0,0)$ along the centerline, given in $\mathcal{P}$ coordinates. For each i , $0 \leq i \leq n$, the **uvn** basis $\mathbf{B}_i$ of the moving frame in the point $\mathbf{p}_i^{\mathcal{P}}(0,0)$ is defined by

$$\mathbf{u} = \mathbf{b}_i \quad , \quad \mathbf{v} = \mathbf{n}_i \quad , \quad \mathbf{n} = \mathbf{t}_i \tag{1}$$

where, in $\mathbf{p}_i^{\mathcal{P}}(0,0)$, $\mathbf{t}_i$ is the normalized tangent to the centerline, $\mathbf{n}_i$ is the unit normal and $\mathbf{b}_i$ the unit binormal. The vectors $\mathbf{t}_i$, $\mathbf{n}_i$ and $\mathbf{b}_i$ are also defined in $\mathcal{P}$ coordinates, and they are computed as described in Section 3.1.1. In every point $\mathbf{p}_i^{\mathcal{P}}(0,0)$, the plane $\mathbf{P}_i$, orthogonal to the centerline, is implicitly defined by $\mathbf{p}_i^{\mathcal{P}}(0,0)$ and $\mathbf{t}_i$ (the normal vector of the plane). Furthermore, let $sector_i$ be the region enclosed between the two planes $\mathbf{P}_i$ and $\mathbf{P}_{i+1}$.

To generate the tubular grid of radius r and resolution s using this moving frame, which creates $\mathcal{T}$, we define the grid points around each $\mathbf{p}_i^{\mathcal{P}}(0,0)$, lying in the plane $\mathbf{P}_i$, as

$$\forall x,y \in \mathbb{Z} \; : \; -s \leq x,y \leq s \; , \; \mathbf{p}_i^{\mathcal{P}}(x,y) = \mathbf{p}_i^{\mathcal{P}}(0,0) + \frac{r}{s}x\mathbf{b}_i + \frac{r}{s}y\mathbf{n}_i \tag{2}$$

The edges of the curvilinear grid are then defined between points $\mathbf{p}_i^{\mathcal{P}}(x,y)$ and $\mathbf{p}_{i+1}^{\mathcal{P}}(x,y)$, and between $\mathbf{p}_i^{\mathcal{P}}(x,y)$ and $\mathbf{p}_i^{\mathcal{P}}(x\pm1,y\pm1)$, forming hexahedral cells (see Figure 4 for an example in 2D). Equation 2 defines a mapping from $\mathcal{S}$ to $\mathcal{P}$; the inverse mapping from $\mathcal{P}$ to $\mathcal{S}$ of a point $[x,y,z]_i^{\mathcal{P}}$ lying on the plane $\mathbf{P}_i$ is defined by

$$[x,y,i]^{\mathcal{S}} = \mathbf{B}_i^{-1}([x,y,z]_i^{\mathcal{P}} - \mathbf{p}_i^{\mathcal{P}}(0,0)) + [0,0,i] \tag{3}$$

Finally, $\mathcal{S}$ is defined by the grid points of $\mathcal{T}$ expressed with respect to their basis $\mathbf{B}_i$, forming a new Cartesian grid, that is the straightened grid $\mathcal{T}$. It should be noted that the space $\mathcal{T}$ is given in $\mathcal{P}$ coordinates, while it is parametrized in $\mathcal{S}$ coordinates. In the following it is sufficient to only consider the two spaces $\mathcal{P}$ and $\mathcal{S}$.

3.1.3 Centerline centric line straightening

With this approach the computation, e.g., by integration, of line primitives, such as streamlines, is performed in the original vector space $\mathcal{P}$. To straighten them, we use an algorithm that creates a parametrization of the points using the local bases from the moving frame. This algorithm performs a piecewise reformation of a line by using the planes $\mathbf{P}_i$, defined by the tangent of the moving frame, as reference (see Figure 5). These planes are defined in a discrete number of equidistant points along the centerline (see Figure 4). From Section 3.1.2 we know how to straighten points that lie in planes $\mathbf{P}_i$, using equation 3. To create a straightened *skeleton* of a line, integrated from a seed point $\mathbf{p}_i^{\mathcal{P}}(r,s)$ lying in plane $\mathbf{P}_i$, we could compute all the intersection points of the line with the planes it intersects during the integration, and then transform these intersections from $\mathcal{P}$ into $\mathcal{S}$.

The following description of the tracing algorithm assumes that the first integration step goes in the direction of $\mathbf{t}_i$, the opposite case is symmetric and we omit a detailed description here. First, let $\mathbf{l}_k$ be the position of the integration front in $\mathcal{P}$ coordinates after k integration steps. In the algorithm, we perform, at each integration step k, an intersection check against the next plane $\mathbf{P}_{i+1}$, and, if it fails, against the current plane $\mathbf{P}_i$, if the sign of the dot-product $[\mathbf{l}_k - \mathbf{l}_{k-1}] \cdot \mathbf{t}_i$ is negative or it is 0. If the dot product is positive we check solely against $\mathbf{P}_{i+1}$. Intersection points are then reformed into $\mathcal{S}$ using equation 3. During the integration, we keep track of the current sector i, containing the integration front, and, at each intersection, we generate a reformed point using the formula described above, and we update the current sector.

Reforming also the points between two consecutive intersection points requires a mean to warp the space between two consecutive planes. Therefore we created a parametrization for points known to lie in a $sector_i$ based on the enclosing planes (illustrated in 2D in Figure 6). Let us assume, for now, that the two planes $\mathbf{P}_i$ and $\mathbf{P}_{i+1}$ are not parallel. Then, assuming that we want to reform the point $\mathbf{p}$, the algorithm can be described as follows:

1. Compute the line $\mathbf{L}$ of the intersection between the two planes $\mathbf{P}_i$ and $\mathbf{P}_{i+1}$.
2. Compute the vectors $\mathbf{lv}_i$ and $\mathbf{lv}_{i+1}$, orthogonal to $\mathbf{L}$, and going from $\mathbf{L}$ to the center points $\mathbf{p}_i^{\mathcal{P}}(0,0)$ and $\mathbf{p}_{i+1}^{\mathcal{P}}(0,0)$ of the quads lying on the two planes. Let $\mathbf{lvn}_i$ and $\mathbf{lvn}_{i+1}$ be the normalized versions of $\mathbf{lv}_i$ and $\mathbf{lv}_{i+1}$.
3. Compute the vector $\mathbf{lv}_p$, orthogonal to $\mathbf{L}$, going from $\mathbf{L}$ to the point $\mathbf{p}$. Compute also the point $\mathbf{lp}$ as the intersection between $\mathbf{L}$ and $\mathbf{lv}_p$. Let $\mathbf{lvn}_p$ be the normalized version of $\mathbf{lv}_p$.
4. Compute the point $\mathbf{q}_i = \mathbf{lp} + \frac{\mathbf{lvn}_i \, |\mathbf{lv}_p|}{\mathbf{lvn}_p \cdot \mathbf{lvn}_i}$. Similarly, compute $\mathbf{q}_{i+1}$.
5. Compute the vector $\mathbf{vq}_i = [\mathbf{q}_i - \mathbf{p}_i^{\mathcal{P}}(0,0)]$. Similarly, compute $\mathbf{vq}_{i+1}$.
6. Transform the vector $\mathbf{vq}_i$ to $\mathcal{S}$, by computing $\mathbf{vq'}_i = \mathbf{B}_i^{-1} \; \mathbf{vq}_i$. Similarly, transform $\mathbf{vq}_{i+1}$.
7. Compute the point $\mathbf{q'}_i = [0,0,i]^{\mathcal{S}} + \mathbf{vq'}_i$. Similarly, compute $\mathbf{q'}_{i+1}$.
8. Compute the reformed point $\mathbf{p}' = \mathbf{q}'_i + [\mathbf{q}'_{i+1} - \mathbf{q}'_i] \; \frac{|[\mathbf{p}-\mathbf{q}_i]|}{|[\mathbf{q}_{i+1}-\mathbf{q}_i]|}$.

If the planes are parallel, it is sufficient to compute $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$ as the intersection of the line $\mathbf{p} + s\mathbf{t}_i$ with the planes $\mathbf{P}_i$ and $\mathbf{P}_{i+1}$ respectively, and then start from point 5. Note that steps 1 and 2 are the same for each point in a $sector_i$, and can, in fact, be precomputed. This approach produces accurate line reformations, meaning that the positions along the reformed line in $\mathcal{S}$ are the reformed positions along the line in the original space $\mathcal{P}$.

3.1.4 Centerline centric vector field reformation

Obtaining a local, and smoothly varying, coordinate frame for every point on a curve allows to perform a straightforward curve centric re-

sampling for scalar volume straightening. However, to reform vector data, it is necessary to transform not only the vectors' magnitude, but in particular their direction. Transforming a vector $\mathbf{u}$, defined in the original space $\mathcal{P}$, into the vector $\mathbf{v}$, defined in $\mathcal{S}$, requires to compute the Jacobian matrix $\mathbf{J}$ for the grid point of the space $\mathcal{S}$ where $\mathbf{u}$ is sampled. $\mathbf{J}$ contains the partial derivatives of the grid (in $\mathcal{P}$ coordinates) with respect to $\mathcal{S}$ in the same point. Let $\mathbf{p}_i^{\mathcal{P}}(x,y)$ be a point in $\mathcal{T}$ expressed in $\mathcal{P}$ coordinates. Then, the Jacobian $\mathbf{J}(x,y,i)$ is defined as

$$\left(\frac{\partial \mathbf{p}_i^{\mathcal{P}}(x,y)}{\partial x} \quad \frac{\partial \mathbf{p}_i^{\mathcal{P}}(x,y)}{\partial y} \quad \frac{\partial \mathbf{p}_i^{\mathcal{P}}(x,y)}{\partial i} \right)$$

As the grid in the space $\mathcal{T}$ is actually defined by our moving frame along the centerline, we know the grid vertices in the neighborhood of $\mathbf{p}_i^{\mathcal{P}}(x,y)$, as they are connected by the edges as defined in Section 3.1.2. In any point of the grid in $\mathcal{T}$, the components $\frac{\partial \mathbf{p}_i^{\mathcal{P}}(x,y)}{\partial x}$ and $\frac{\partial \mathbf{p}_i^{\mathcal{P}}(x,y)}{\partial y}$ are given by the vectors $\mathbf{b}_i$ and $\mathbf{n}_i$. Thus, the only component that has to be estimated to build the matrix $\mathbf{J}(x,y,i)$ is $\frac{\partial \mathbf{p}_i^{\mathcal{P}}(x,y)}{\partial i}$. This term can be approximated, for example, by using one of three differential operators: central differences, forward differences or backward differences, as described also by Sadarjoen et al. [29]. For our purposes, we use a mixed forward and backward difference operator, depending on the sign of the dot product of the vector $\mathbf{v}_i^{\mathcal{P}}(x,y)$, sampled in the point $\mathbf{p}_i^{\mathcal{P}}(x,y)$, with the normal $\mathbf{n}_i$. This way we introduce less smoothing, compared to using central differences.

We therefore modify the CCVR method [12] to handle vector data, using the following equation

$$\mathbf{vec}_i^{\mathcal{S}}(x,y) = \mathbf{J}^{-1}(x,y,i)\mathbf{vec}_i^{\mathcal{P}}(x,y) \tag{4}$$

If $\mathbf{p}_i^{\mathcal{P}}(x,y)$ is not a grid point of the data space $\mathcal{P}$, $\mathbf{vec}_i^{\mathcal{P}}(x,y)$ has to be reconstructed using an interpolation scheme. In case of vector data this operation can be done in different ways. The simplest approach is to perform a per-component trilinear interpolation. However, in case of a vector field, this might not be the best solution, as it linearly interpolates only the direction of the vectors, not the length. A different approach, that we adopt in our prototype, consists of using spherical linear interpolation (slerp) using quaternions, in order to interpolate also vector lengths.

This method generates a straightened vector field, and primitive integration as well as other flow visualizations can be performed directly in $\mathcal{S}$, without the need of subsequent reformation. However, due to numerical inaccuracies, this approach and the one presented in Section 3.1.3 might not lead to identical results. We have compared this approach with the one described in Section 3.1.3, and the results are presented in Section 4.3

3.2 Side-by-side straightened flow visualizations

By *complementing* the visualization of the original data with a juxtaposition of views of the straightened tubular flow, we aim to, first, provide a common axis for the alignment and co-registration of different views of the data, to easily relate them to each other. This also allows to create compact visualizations that give good overviews of the data, even combining different visualization techniques. Third, we want to statically convey the data variations over time, in case of unsteady flow datasets. Last, we want to help comparing different aspects of a dataset (such as different time points, or different descriptors), or even different datasets (as in population studies). In the next Section we describe a set of techniques to handle such straightened views properly, in order to create side-by-side straightened visualizations that fully exploit the possibilities that this method offers.

3.2.1 Visualization design and layout

When assembling the visualization, particular attention must be put in the combination of the views. First, the straightening axis in the views should be aligned to one of the screen axes, in order to facilitate the juxtaposition and the alignment of several views. Having the reformed centerline aligned to one of the screen axes also allows to minimize the space in between different views. This alignment also allows to combine visualization of the actual data (such as standard flow visualization techniques) with more abstract visualization techniques, such as a line graph plotting certain quantities along the centerline (see Figure 7(a,b)). In such setup, it becomes possible to use the centerline axis as navigational tool: it can be used for operations such as cross-section placement and movement, and length measurement (see Figure 2).

Second, special attention must be put in conveying the shape of the reformation, in order to enable the viewer to easily relate positions and directions in the reformed view to positions and directions in the original space. We propose to use two kind of orientation cues. The primary cue is the rendering of the reformed tubular structure around the flow as its spatial context. To do this, we perform volume ray casting of the straightened context, instead of rendering the extracted isosurface. This allows us to perform fast and correct depth-buffer based alpha blending with the integrated geometric primitives, such as streamlines, in a single (modified) ray casting pass, without the need of performing expensive multi-pass rendering techniques, such as depth-peeling. In addition, we propose to use a number of "i-shaped" glyphs along the projection of the normal and of the binormal onto the flow

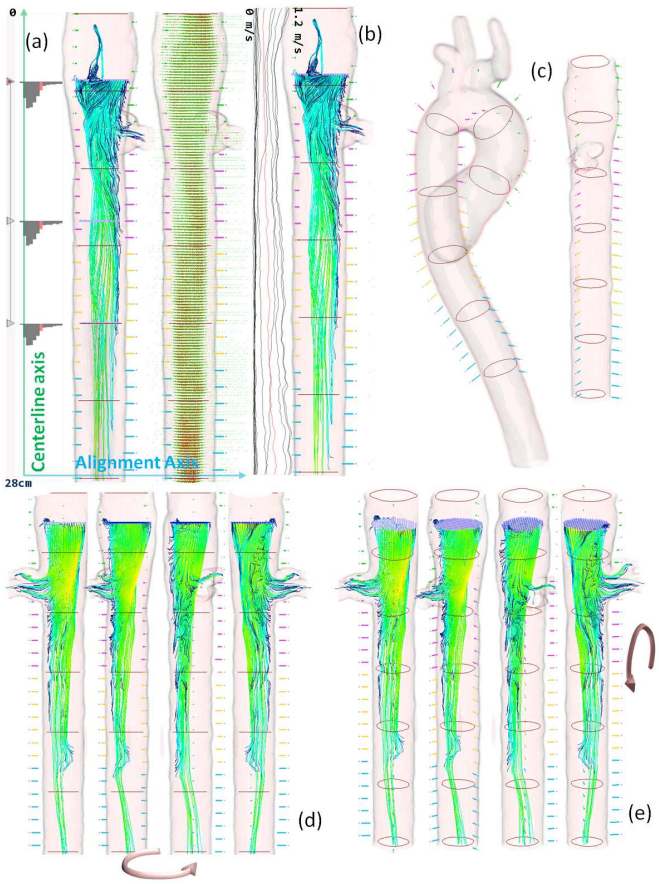

Fig. 7. Design approaches to a side-by-side visualization of straightened tubular flow: straightened views should be aligned to one of the screen axes, and juxtaposed along the other. The first axis also serves to place navigational widgets to interact with the visualization along the centerline. The second axis is used to relate different views to each other. Informative visualizations, such as a line graph or a histogram of the flow magnitude can also be placed along the centerline, to provide quantitative information (a,b). Orientation cues are needed for orientation: we use volume rendering of the physical context, with contours, to convey the physical space. For additional orientation cues, we add glyphs (c). Interaction with the visualization should be modified to allow only meaningful camera transformations. We use only rotations around the two axes used for the alignment (d,e).

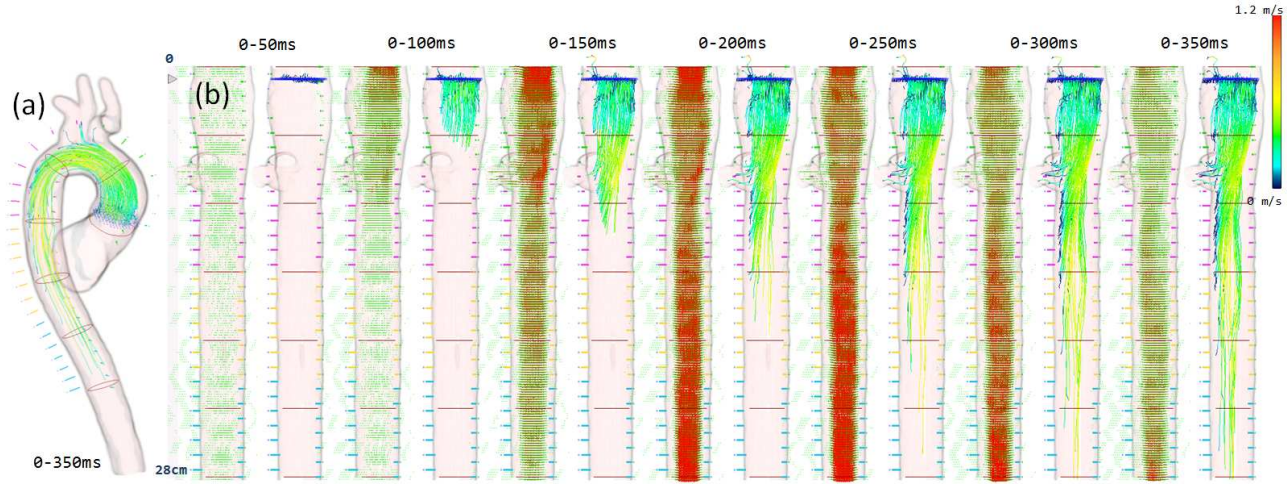

Fig. 8. Timesteps from 0 to 6 of the aorta dataset, visualized with pathlines and glyphs illustrating the vector field. Each pair of views shows the glyphs rendering of the vector field at the last time point stated on top, and the pathline integration from time 0 to the last time point.

bounding structure (see Figure 7). The body color of these glyphs encodes the distance from the beginning of the centerline, while the dot color encodes the projection axis (green dot = glyph above the normal, blue dot = glyph above the binormal). In this way we help the user to orient and understand from which viewpoint she is looking at the flow. These glyphs, in combination with a specified number of isocontours of the tubular structure, also help the user relating a region along the centerline "axis" between the conventional view and the reformed side-by-side visualization.

The proposed side-by-side layout also introduces some challenges in the interaction process with the visualization. Rotating the visualization with the classical joystick or trackball paradigms, in particular, might become unfeasible. For this reason we enable only 2 rotation methods: per-view rotation around the centerline axis, and global rotation around the other screen axis (see Figure 7). This method proved to allow thorough exploration of the straightened data, while, at the same time, being intuitive and error-proof, preventing that the user might "get lost" while interacting with the visualization.

3.2.2 Straightened side-by-side visualization

In this section we illustrates some of the visualization opportunities offered by juxtaposing straightened flow views. The most obvious opportunity is to visualize many timesteps of an unsteady flow at the same time, aligned along the same axis, as shown in Figure 2. In this way it is possible to convey the temporal evolution in one single, compact visualization, that also allows to immediately relate the same region (position along the centerline) of the flow in different timesteps.

Another possibility consists of generating a compact, thorough view of the flow from different angles (see Figure 7(d,e)). This is particularly useful when inspecting tubular structures in complex shapes, for which few projections might still not make all the flow content visible. With only few views of the straightened flow from equiangular view points it becomes possible to inspect the flow from all possible sides.

Finally, this technique permits the juxtaposition of different types of visualizations side by side and relate them with each other. As an example, in Figure 8 we show a composition of pairs of visualizations, showing pathlines at each timestep next to the representation of the vector field at the same timestep. The clear advantage is again the simplicity of spatially relate the different aspects of the same data (the timestep). In Figure 2(right) different aspects of the same timestep (streamlines integration from different seeding planes) are also placed side by side, highlighting the contribution of each seeding plane to the result, on the left side.

4 Realization and Evaluation

To use our technique, we developed a prototype, making partly use of VTK. In the prototype we implemented streamline and pathline tracing on the GPU, using geometry shaders [31]. We also implemented abstract visualization techniques such as line graphs and histograms (of averaged velocity), to demonstrate the simplicity of combining classic flow visualizations with other data visualization methods in an intuitive way. The proposed approach can, however, be also used with other types of flow visualization. We used our prototype to successfully visualize two datasets from different fields, which we describe in the next sections.

4.1 Aortic flow visualization

Magnetic Resonance Imaging (MRI) is one of the fastest developing imaging technologies in medicine. Recently, improved time-resolved 3D Phase-Contrast MRI (PC-MRI) has been successfully used to image the blood flow in the human body. Bock et al. [1] provide an overview of this imaging modality describing the characteristics of the generated data. The dataset we visualize is a vector field of a human aorta, specified on a Cartesian grid with a resolution of $192 \times 144 \times 24$ voxels in x, y and z respectively, containing 13 time steps acquired at a time resolution of about 50 milliseconds. The spatial resolution of the scan is [1.67mm, 1.67mm, 3.5mm] in x, y and z, for an imaged volume of $32 \times 24 \times 8.5$ cm. To simplify the handling of the significant anisotropy of this dataset, we decided to upscale the dataset to an isotropic grid beforehand. The aorta was segmented as described in Section 3, and the computed centerline of the arterial wall measured about 30cm. The centerline was subdivided in segments of voxel-length to minimize resampling artifacts, and the straightening was performed using quads with a side length of approximately 7.5 cm, with a transversal resampling resolution of 49×49 voxels (approximately the same resolution of the data). Figure 1 shows all the timesteps side by side using streamlines with a fixed seeding grid and 3 seeding planes, presenting the whole time-lapse with static time dependency. In Figure 2 we investigate a single timestep, by separating the seeding body into different views, to prevent streamlines overlapping. Finally, In Figure 8 we show the evolution of pathlines integration, from timestep 1 to 7, together with the vector field at each timestep. In this way we effectively combine different methods in a side-by-side visualization of the flow.

4.2 Exhaust system flow visualization

This dataset contains the simulation results of an exhaust system with 3 collectors from the cylinders and a common rail for the emission. The dataset is a vector field specified on a Cartesian grid, with resolution of $133 \times 82 \times 68$ voxels in x, y and z respectively, over 30

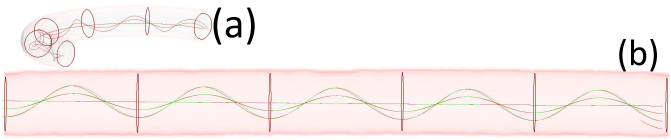

Fig. 9. Synthetic dataset, a curved tube containing helical flow, with four streamlines seeded at the beginning of the tube, along the radius (a). The same dataset, straightened, with streamlines integrated in the reformed flow (red) and in the original flow and then deformed (green) (b). The average distance between the pairs of lines is 0.10 voxel, while the maximum distance is 0.21 voxel.

time steps. We computed the centerline starting at the beginning of the first collector to the end of the rail, thus analyzing the behavior of this part of the system. The centerline was subdivided again in segments of voxel-length, and the straightening was performed using quads of radius 20 voxels. In Figure 10 we visualize a time lapse of the flow, from timestep 0 to timestep 17, using streamlines, traced from 3 seed planes placed after each collector. The image clearly conveys the valves opening sequence (ts 1 = 2, ts 6 = 1, ts 14 = 3), and the curve of the decreasing velocity after the closure.

4.3 Performance and Error Analysis

We have compared the performance of standard streamline integration performed on the reformed vector field with the approach presented in Section 3.1.3, using a CPU implementation of both algorithms on an Intel Core2 2.4ghz processor. We seeded 1000 streamlines on plane $\mathbf{P}_0$, the beginning of the centerline, on both the aorta and the manifold dataset, and used 1000 integration steps, with a step size of 0.25 voxel, on 10 different timesteps of each dataset. After averaging of the results, the standard integration took 0.65 seconds to complete this task seconds, while the line straightening method required 3.75 seconds. The second algorithm also showed higher variance in the results. This behavior can be explained considering that some timesteps contains low velocities, and the integration crosses only few sectors along the centerline, lowering the computational complexity. The conclusion is that the accurate approach is about 6 times slower than the other one.

We then carried out an error analysis to compute the average and maximum gap between lines traced with the two approaches, when they are seeded at the exact same locations. We measured the error by stepping along each pair of lines (the one integrated in the reformed field and the one straightened), using a step size of 0.25 voxel. At each step, we mesured the distance between the corresponding locations along the two lines. The table below reports the average and maximum distance (expressed in voxel units) for the different datasets, averaged over 1000 streamlines and traced on 10 different timesteps. For this analysis we also added a synthetic dataset proposed by Roth and Peikert [28] (see Figure 9), consisting of a helical flow inside a bent pipe, for which the centerline is known. This analysis shows that the measured PC-MRI dataset is the one where the vector field reformation leads to the largest error. One of the reasons could be that the extracted centerline is not 100% accurate, and therefore the cross planes do not result perfectly orthogonal to the vessel. This may lead to inaccurate Jacobian computation for the two transversal components, that are taken directly from the moving frame. We can conclude that a crucial aspect of our technique is a robust and accurate centerline extraction algorithm, to be able to accurately integrate the reformed vector field.

	Aorta	Exhaust Manifold	Bent pipe
Average Diameter	23	24	18
Average Error	1.17	0.39	0.33
Maximum Error	1.83	1.07	0.72

Table 1. Average and maximum distance between streamlines integrated in the reformed vector field and streamlines straightened after the integration in the original field. The values are expressed in voxels.

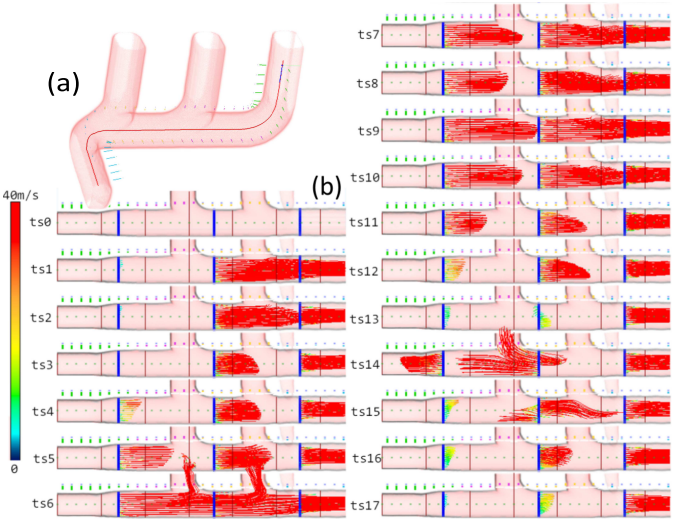

Fig. 10. The exhaust system, volume rendering (a) Static time lapse visualization of the straightened flow in the exhaust system dataset, timesteps from 0 to 17 (b).

4.4 Evaluation

The Cardiovascular MRI Group at the University Medical Center Freiburg, Medical Physics department, very kindly provided us with an informal evaluation of the presented technique, that we demanded in order to understand how possible end users would benefit from it. This evaluation is composed of general impressions and of answers to specific questions we asked. In general, the reformatting of the aorta has been seen as potentially useful to compare some hemodynamic parameters (such as wall shear stress or pressure differences), also across a population. However, in this case one would need some kind of aortic atlas, and then map the dataset onto this atlas (a starting point for this mapping could be actually found in the work of Ropinski et al. [27]). There was also some uncertainty about how the visualization would look in presence of an aneurysm or a stenosis. The group also believe that medical personnel is more accustomed to seeing the blood flow in its original context, and would, therefore, require a certain training in order to profit from the proposed method.

The specific questions we asked to the Cardiovascular MRI Group in Freiburg were what kind of visual comparison are they interested in, whether this approach would ease the comparison of integrated lines in the aorta, and what are other parameters typically investigated. Then we asked whether they think that physicians would profit from this technique as well, and what do physicians generally look at, in such data. According to their answers, at the present they do not perform that much comparison visually, but the presented approach could be useful to compare hemodynamic parameters, while other typical parameters of interest along the vessel are helicity and vorticity. The presented approach has been seen as definitely easing the comparison of integrated lines from their point of view, but, from a medical point of view, pysicians are currently very accustomed to the original shape of the vessel. Last, visual features of interest from the medical point of views are helices, vortices, and retrograde flow at late timepoints.

From this evaluation we can conclude that domain experts could profit from this flow straightening techniques, but some training is necessary. However, there have been other cases of reformation techniques which required a certain learning, before being embraced in the clinical routine, such as the curved-planar reformation of the human vessel tree [10].

5 Summary and Conclusions

In this paper we present a general solution for producing straightened tubular flow views by applying standard flow visualization techniques to a straightened vector field along the centerline of the tubular object. In addition, we presented multiple techniques for composing such

views, in order to form straightened side-by-side visualizations. We used our method to visualize two different tubular flow datasets, showing that the technique is generally applicable for any dataset where the flow under inspection streams within a tubular structure. With the generated side-by-side visualizations we achieved improvements over standard techniques, in terms of efficiency in the usage of the available visualization space, and in terms of ease in the comparison of the different aspects of the data. We received a positive feedback by domain experts, that let us conclude that it is worthwhile, in certain cases, to choose an alternative way to look at the data over the conventional ones, to exploit the power of visualization. Limitations of our approach are, at the present, the handling of structures with unnatural narrow bends, when cross planes intersects each other within the lumen of the pipe, and the handling of structures with major bifurcations. Both of these issues require further investigations. As a future work, we also plan to investigate more thoroughly the perception of flow straightening for longer, more complex structures and to obtain a more formal evaluation.

ACKNOWLEDGMENTS

We are very grateful to Jelena Bock and to all the Cardiovascular MRI Group, Medical Physics at the University Medical Center, Freiburg, for the PC-MRI dataset and the very valuable feedback provided. The exhaust manifold dataset is courtesy of AVL GmbH, Graz, Austria, and we thank Alan Lež, Krešimir Matković and Philipp Muigg for helping us to import the data. Furthermore, we want to thank Andrea Brambilla, Ove Daae Lampe, Július Parulek, Armin Pobitzer and Eirik Vik for all the fruitful discussions together. Parts of this work originate in the EC-funded SemSeg project (FET-Open, grant number 226042), with the support of the MedViz network in Bergen, Norway (PK1760-5897-Project 11).

REFERENCES

[1] J. Bock, A. Frydrychowicz, A. Stalder, T. Bley, H. Burkhardt, J. Hennig, and M. Markl. 4D phase contrast MRI at 3 T: Effect of standard and blood-pool contrast agents on SNR, PC-MRA, and blood flow visualization. *Magnetic Resonance in Medicine*, 63(2):330–338, 2010.

[2] M. Borkin, S. Melchionna, C. Feldman, E. Kaxiras, and H. Pfister. Multidimensional visualization of hemodynamic data. Proceedings of IEEE Visualization Conference, Atlantic City, USA, October 2009.

[3] M. Chen, D. Silver, A. Winter, V. Singh, and N. Cornea. Spatial transfer functions: a unified approach to specifying deformation in volume modeling and animation. In *Proceedings of the 2003 Eurographics/IEEE TVCG Workshop on Volume graphics*, pages 35–44. ACM, 2003.

[4] N. Cornea, D. Silver, and P. Min. Curve-skeleton properties, applications, and algorithms. *Visualization and Computer Graphics, IEEE Transactions on*, 13(3):530–548, 2007.

[5] C. Correa, D. Silver, and M. Chen. Discontinuous displacement mapping for volume graphics. In *Proceedings of Volume Graphics*, volume 6, pages 9–16, 2006.

[6] H. Doleisch. SIMVIS: interactive visual analysis of large and time-dependent 3D simulation data. In *Proceedings of the 39th conference on Winter simulation*, pages 712–720. IEEE Press, 2007.

[7] F. Frenet. Sur les courbes a double courbure. *Journal des Mathematiques Pures et Appliquees*, 17:437–447, 1852.

[8] R. Fuchs and H. Hauser. Visualization of Multi-Variate Scientific Data. In *Computer Graphics Forum*, volume 28, pages 1670–1690. Wiley Online Library, 2009.

[9] C. Jones and K. Ma. Visualizing Flow Trajectories Using Locality-based Rendering and Warped Curve Plots. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1587–1594, 2010.

[10] A. Kanitsar, R. Wegenkittl, D. Fleischmann, and M. Gröller. Advanced curved planar reformation: Flattening of vascular structures. *Visualization and Computer Graphics, IEEE Transactions on*, pages 43–50, 2003.

[11] R. Kirby, D. Keefe, and D. Laidlaw. Painting and visualization. *The Visualization Handbook*, pages 873–891, 2005.

[12] O. Lampe, C. Correa, K. Ma, and H. Hauser. Curve-centric volume reformation for comparative visualization. *IEEE Transactions on Visualization and Computer Graphics*, pages 1235–1242, 2009.

[13] R. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. Post, and D. Weiskopf. The State of the Art in Flow Visualization: Dense and Texture-Based Techniques. In *Computer Graphics Forum*, volume 23, pages 203–221. Wiley Online Library, 2004.

[14] R. Laramee, H. Hauser, L. Zhao, and F. Post. Topology-based flow visualization, the state of the art. *Topology-based Methods in Visualization*, pages 1–19, 2007.

[15] T. C. Lee, R. L. Kashyap, and C. N. Chu. Building skeleton models via 3-d medial surface/axis thinning algorithms. *CVGIP: Graph. Models Image Process.*, 56:462–478, November 1994.

[16] D. Lesage, E. Angelini, I. Bloch, and G. Funka-Lea. A review of 3d vessel lumen segmentation techniques: Models, features and extraction schemes. *Medical Image Analysis*, 13(6):819–845, 2009.

[17] A. Lez, A. Zajic, K. Matkovic, A. Pobitzer, M. Mayer, and H. Hauser. Interactive exploration and analysis of pathlines in flow data. In *Proc. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2011)*, pages 17–24, 2011.

[18] M. Markl, P. Kilner, and T. Ebbers. Comprehensive 4D velocity mapping of the heart and great vessels by cardiovascular magnetic resonance. *Journal of Cardiovascular Magnetic Resonance*, 13(1):7, 2011.

[19] T. McLoughlin, R. Laramee, R. Peikert, F. Post, and M. Chen. Over Two Decades of Integration-Based, Geometric Flow Visualization. In *Computer Graphics Forum*, volume 29, pages 1807–1829. Wiley Online Library, 2010.

[20] S. Meier, A. Hennemuth, O. Friman, J. Bock, M. Markl, and T. Preusser. Non-invasive 4D Blood Flow and Pressure Quantification in Central Blood Vessels via PC-MRI. *Computing in Cardiology*, 37:903–906, 2009.

[21] T. Nobrega, D. Carvalho, and A. von Wangenheim. Simplified simulation and visualization of tubular flows with approximate centerline generation. In *Computer-Based Medical Systems*, pages 1–7. IEEE, 2009.

[22] H. Pagendarm and F. Post. Studies in comparative visualization of flow features. *Scientific Visualization, Overviews, Methodologies, and Techniques*, pages 211–227, 1997.

[23] Z. Peng and R. Laramee. Higher Dimensional Vector Field Visualization: A Survey. *Theory and Practice of Computer Graphics (TPCG09)*, pages 149–163, 2009.

[24] A. Pobitzer, R. Peikert, R. Fuchs, B. Schindler, A. Kuhn, H. Theisel, K. Matkovic, and H. Hauser. On the way towards topology-based visualization of unsteady flow - the state of the art. In *EuroGraphics 2010 State of the Art Reports (STARs)*, pages 137–154, 2010.

[25] F. Post, B. Vrolijk, H. Hauser, R. Laramee, and H. Doleisch. Feature extraction and visualization of flow fields. *Eurographics 2002 State-of-the-Art Reports*, pages 69–100, 2002.

[26] F. Post, B. Vrolijk, H. Hauser, R. Laramee, and H. Doleisch. The state of the art in flow visualisation: Feature extraction and tracking. In *Computer Graphics Forum*, volume 22, pages 775–792. Wiley Online Library, 2003.

[27] T. Ropinski, S. Hermann, R. Reich, M. Schäfers, and K. Hinrichs. Multimodal vessel visualization of mouse aorta PET/CT scans. *IEEE Transactions on Visualization and Computer Graphics*, pages 1515–1522, 2009.

[28] M. Roth and R. Peikert. Flow visualization for turbomachinery design. In *Proceedings of the 7th conference on Visualization'96*, pages 381–384. IEEE Computer Society Press, 1996.

[29] A. Sadarjoen, T. Van Walsum, A. Hin, and F. Post. Particle tracing algorithms for 3D curvilinear grids. *Scientific Visualization, Overviews, Methodologies, and Techniques*, pages 311–335, 1997.

[30] T. Salzbrunn, H. Jänicke, T. Wischgoll, and G. Scheuermann. The state of the art in flow visualization: Partition-based techniques. *Simulation and Visualization 2008 Proceedings*, pages 75–92, 2008.

[31] R. van Pelt, J. Bescós, M. Breeuwer, R. Clough, M. Gröller, B. ter Haar Romenij, and A. Vilanova. Exploration of 4D MRI Blood Flow using Stylistic Visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1339–1347, 2010.

[32] V. Verma and A. Pang. Comparative flow visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 10(6):609–624, 2004.

[33] A. Vilanova, E. Gröller, and A. König. Cylindrical approximation of tubular organs for virtual endoscopy. In *Proceedings of Computer Graphics and Imaging*, volume 11, pages 283–289, 2000.

[34] A. Vilanova, R. Wegenkittl, A. Konig, and E. Gröller. Nonlinear virtual colon unfolding. *Visualization and Computer Graphics, IEEE Transactions on*, pages 411–418, 2001.

[35] T. Vrtovec, B. Likar, and F. Pernuš. Automated curved planar reformation of 3D spine images. *Physics in medicine and biology*, 50:4527, 2005.