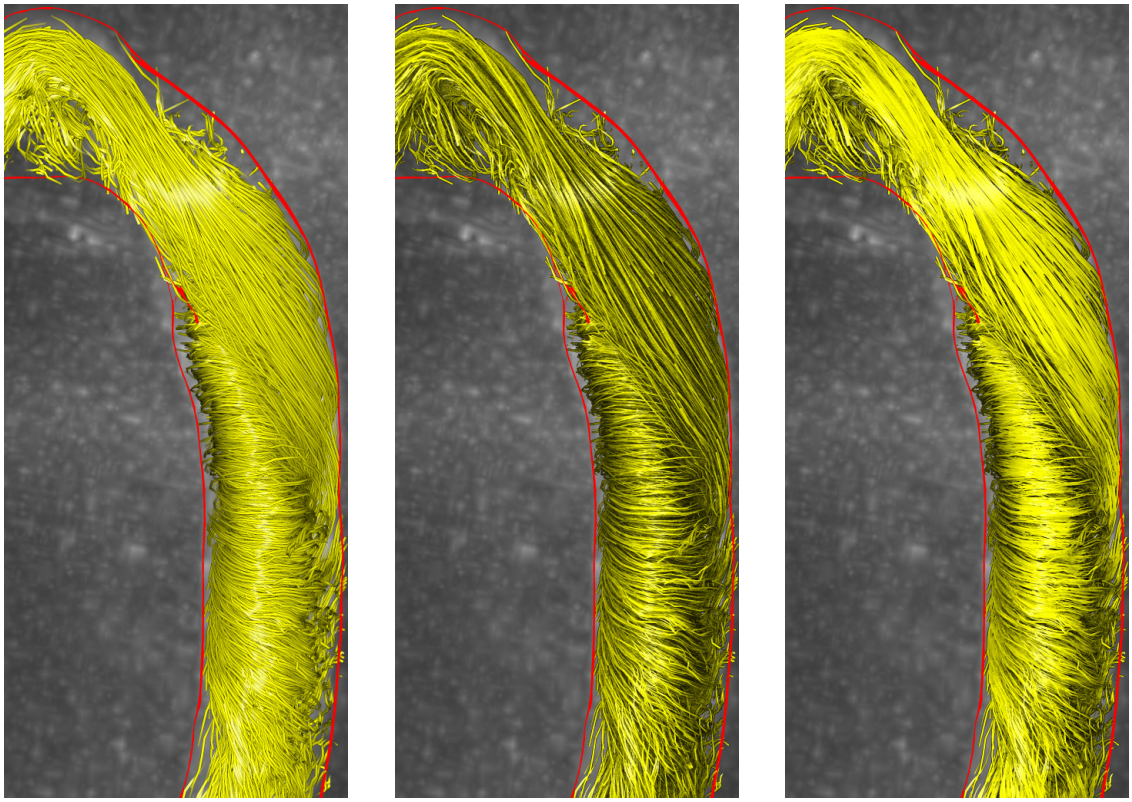
Vision, Modeling, and Visualization (2017)
M. Hullin, R. Klein, T. Schultz, A. Yao (Eds.)

Visualization of Cardiac Blood Flow Using Anisotropic Ambient Occlusion for Lines

Benjamin Köhler[1], Matthias Grothoff[2], Matthias Gutberlet[2], and Bernhard Preim[1]

[1] Otto-von-Guericke University, Magdeburg, Germany [2] Heart Center, Leipzig, Germany



(a) *Reference* **(b)** *LineAO [EHS13]* **(c)** *Our anisotropic LineAO*

Figure 1: *Descending aorta of a patient with an aortic arch prosthesis. (a) Reference: Line illumination and uniform halos. (b) Ambient occlusion for lines (LineAO) [EHS13]. (c) Our proposed anisotropic LineAO.*

Abstract

Ambient occlusion (AO) for lines (LineAO) was introduced by Eichelbaum et al. [EHS13] as an adaption of screen-space AO to static line bundles, such as white brain matter fiber tracts derived from diffusion tensor imaging (DTI). In this paper, we further adapt the LineAO technique to dynamic scenes, in particular the animation of blood flow-representing pathlines that were integrated in cardiac 4D phase-contrast magnetic resonance imaging (PC-MRI) data. 4D PC-MRI is a non-invasive technique that allows to acquire time-resolved blood flow velocity data in all three spatial dimensions, i.e., a 4D vector field of one heart beat. Our main extension is a line alignment factor that reduces the AO-induced darkening if nearby lines have similar screen-space tangents. We further enhance the perception of homogeneous flow by incorporating depth-dependent halos. Our technique facilitates the quicker identification of prominent flow structures while showing the full flow context.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture, J.3 [Computer Applications]: Life and Medical Sciences—

1. Introduction

4D phase-contrast magnetic resonance imaging (PC-MRI) [SAG*14] is a non-invasive method that allows to measure time-resolved blood flow directions of one heart beat. The integration of pathlines in combination with standard flow visualization techniques, like line illumination or halos (contours) as well as their display in animations, are the most common approaches to visualize cardiac 4D flow data. Medical researchers usually obtain a first impression of the flow behavior by watching an animation of the full flow (whole vessel covered by pathlines). However, distinct characteristics may easily be missed due to the high complexity of the 4D PC-MRI data. This led to the development of custom feature extraction methods [KGP*13, MKP*16] and simplifications [BMGS13] that hide irrelevant context.

Spatial relations can be challenging to interpret in visualizations of dense, blood flow-representing pathlines. Lights and shadows are essential for the spatial perception of any scene [Ram88, Wan92, WFG92, LB00]. In the last decades, different shading techniques were developed with the goal to render a scene as realistically as possible. Some methods are real-time capable, others provide better results at higher computational costs. Lines, however, require special treatment. They do not have a unique normal vector, which is essential for most methods. In addition, there is a vast amount of intersections and crossings compared to natural scenes.

Ambient occlusion (AO) for lines (LineAO) – an adaption of screen-space AO (SSAO) for line bundles – was introduced by Eichelbaum et al. [EHS13] as advanced illumination technique to attenuate line segment colors based on local depth differences. This technique was tailored towards white brain matter fiber tracts derived from diffusion tensor imaging (DTI). These are static streamline bundles without a temporal component. In this work, we propose anisotropic LineAO for the dynamic cardiac context and make the following contributions:

- We analyze the suitability of LineAO for dynamic scenes, in our case blood flow-representing pathlines.
- We add a new line alignment factor to the LineAO calculation that attenuates the AO effect in case of similar line tangents.
- We reason why depth-dependent halos are a valuable addition in combination with our line alignment factor.

2. Background

2.1. Flow Visualization

Following the classification by Post et al. [PVH*03], techniques can be divided into geometry-, feature-, and texture-based methods. The first group is relevant for this work. Starting from specific seed points, geometry-based methods calculate flow trajectories that, in the time-dependent case, depict the course of a particle in the flow. Such a pathline integration is commonly performed with a scheme from the Runge-Kutta family [DP80].

Illumination is essential for the spatial perception of 3D scenes. If pathlines are rendered as geometric tubes or tapes, line illumination is directly applicable due to existing normal vectors. If they are rendered as actual lines, there exists an infinite number of normal vectors in a circle around the line. Line illumination can then be achieved by selecting consistent normal vectors that depend on the normalized tangent $\vec{T}$ and normalized light direction $\vec{L}$ as $(\vec{T} \times \vec{L}) \times \vec{T}$ [ZSH96].

Halos (line contours) [IG97] enhance the perception of crossing lines' occlusion relations. The idea is that the contour of the line in the front is continuous, whereas the one of the line in the back is interrupted. According to Preim et al. [PBC*16], such techniques work well for sparse datasets, but not so well for dense sets of lines with unstructured orientations. Everts et al. [EBRI09] proposed depth-dependent halos where lines with minor depth differences get a small or no halo at all and higher depth differences result in prominent halos.

Günther et al. presented an opacity optimization for 3D line sets [GRT14] that was extended to combinations of points, lines, and surfaces [GTG17]. They require an importance measure $\in [0,1]$ that is processed in an order-independent transparency (OIT)-based setup on the GPU [YHGT10]. A problem of transparency modification is that it can negatively influence depth perception [BBD*09].

Hair rendering is a field of application where realistic illumination of extremely dense line sets is pursued. Such techniques often use geometric simplifications to reduce the computational burden. Yuksel et al. [YT10] provide an overview.

2.1.1. Ambient Occlusion

Ambient occlusion (AO) [Lan02] is a light-independent, global, physics-based illumination technique that supports the perception of complex 3D structures [PBC*16]. The basic idea is to darken parts of a surface depending on its local occlusion. Therefore, rays are sent into the hemisphere around the surface normal, and for each of them it is evaluated whether or not they intersect another object, i.e., if they are occluded. The number of rays and the radius of the hemisphere are parameters. The percentage AO $\in [0,1]$ of occluded rays is then multiplied as $1 - \text{AO}$ to darken the color.

AO can be calculated as a pre-processing step in static scenes, from which follows that scene changes require recalculation. For dynamic scenes, the real-time capable screen-space ambient occlusion (SSAO) was developed – a standard technique used in many of today's applications, such as games [Mit07]. The idea is to employ a deferred shading setup where information are stored in a framebuffer object (FBO) with multiple textures (G-buffer, also deep framebuffer) in a first rendering pass. The actual lighting etc. is performed in the second pass. Hence, deferred shading depends more on the number of pixels on the screen instead of the scene's geometric complexity (number of triangles). SSAO samples depth values and normal vectors within a circle around the current pixel instead of evaluating rays on a surface normal's hemisphere.

LineAO: Dense line bundles are a special kind of scene that commonly occurs in flow visualization. Here, default SSAO does not produce optimal results. Eichelbaum et al. [EHS13] addressed this problem by introducing LineAO – a SSAO variant tailored towards such scenes. During G-buffer creation in the first deferred shading rendering pass they store:

- A RGB *color map* of the fragment colors,

- a RGBA *ND map* where RGB is the normal vector [ZSH96, MPSS05] and A is the linearized depth, and
- a luminance *zoom map* that holds the zoom level $z(\vec{P})$, which is the actual diameter of a unit sphere at position $\vec{P}$ in screen-space.

In the second pass, which draws a screen-filling quad, the LineAO value is calculated as:

$$\mathrm{LineAO}_{s_r,s_h,r_0}(\vec{P}) \;=\; \sum_{l=0}^{s_r-1} \mathrm{AO}_{\frac{s_h}{l+1},j}\big(\vec{P}, r_0 + l\cdot z(\vec{P})\big) \qquad (1)$$

The authors introduced a special sampling scheme, where s_r radii in s_r texture mipmap levels l (Gaussian pyramid) of the ND map are processed. The authors recommend $s_r = 3$ as default, i.e., the original texture and two smaller versions are needed. They generate the mipmaps manually in order to avoid the computation of unneeded levels, as done by *glGenerateMipMap*, which generates the whole Gaussian pyramid. The first radius r_0 should be slightly wider than a line. As default, the authors suggest $r_0 = 1.5 \cdot$ line width $\cdot\, z(\vec{P})$ for tubes, which is also applicable to lines rendered as view-aligned quads. Each further radius r adds a value to r_0 depending on the mipmap level and zoom value. s_h is the number of samples within the first radius. The suggested default is 32. With increasing mipmap level, this number decreases as $s = s_h/(l+1)$. Using the standard settings, 59 samples per pixel are evaluated. The AO function is defined as:

$$\mathrm{AO}_{s,l}(\vec{P}, r) \;=\; \frac{1}{s}\cdot\sum_{i=1}^{s}\Big[\big(1 - V_l(r\cdot\vec{\omega}_i,\vec{P})\big)\cdot g_l(r\cdot\vec{\omega}_i,\vec{P})\Big] \qquad (2)$$

The binary visibility function $V_l(r\cdot\vec{\omega}_i,\vec{P})$ determines if a pixel is occluded by comparing its linearized depth value $d_l(\vec{P})$ to the one of the current neighbor sample $d_l(\vec{P}+\vec{\omega})$:

$$V_l(r\cdot\vec{\omega},\vec{P}) \;=\; \begin{cases} 0, & \text{if } d_l(\vec{P}) - d_l(\vec{P}+\vec{\omega}) < 0 \\ 1, & \text{else.} \end{cases} \qquad (3)$$

The occlusion per sample $g_l(\vec{\omega},\vec{P})$ on the hemisphere is a product of two factors:

$$g_l(\vec{\omega},\vec{P}) \;=\; g_l^{depth}(\vec{\omega},\vec{P}) \;\cdot\; g_l^{light}(\vec{\omega},\vec{P}) \qquad (4)$$

The first factor $g_l^{depth}(\vec{\omega},\vec{P})$ addresses the depth differences of the current point $\vec{P}$ and the evaluated sample $\vec{P}+\vec{\omega}$. If $\vec{P}$ is occluded, the value increases with higher depth differences:

$$g_l^{depth}(\vec{\omega},\vec{P}) \;=\; \begin{cases} 0, & \text{if } \Delta d_l(\vec{\omega},\vec{P}) > \delta(l) \\ 1, & \text{if } \Delta d_l(\vec{\omega},\vec{P}) < \delta_0 \\ 1 - h\big(\frac{\Delta d_l(\vec{\omega},\vec{P}) - \delta_0}{\delta(l) - \delta_0}\big), & \text{else.} \end{cases} \qquad (5)$$

$\delta_0 = 0.0001$ (authors' fixed parameter) as well as $\delta(l)$ and $h(x)$ are used to fine-tune the resulting depth attenuation:

$$h(x) \;=\; 3\cdot x^2 - 2\cdot x^3,\ \forall x \in [0,1] \quad \text{(Hermite polynomial)} \qquad (6)$$

$$\text{and} \quad \delta(l) \;=\; \Big(\frac{1-l}{s_r}\Big)^2 \qquad (7)$$

The aim of the second factor $g_l^{light}(\vec{\omega},\vec{P})$ is to avoid that occluded parts of the scene that are directly lit by a light source appear dark. The AO value is diminished in such cases:

$$g_l^{light}(\vec{\omega},\vec{P}) \;=\; 1 - \min(L_l(\vec{\omega},\vec{P}),\ 1) \qquad (8)$$

$$\text{with} \quad L_l(\vec{\omega},\vec{P}) \;=\; \sum_{s\in\text{Lights}} \mathrm{BRDF}\big(\vec{L}_s,\ I_s,\ \vec{n}_l(\vec{P}),\ \vec{\omega}\big) \qquad (9)$$

$L_l(\vec{\omega},\vec{P})$ is the reflected light at $\vec{P}$ in direction of $\vec{\omega}$. We use head light with intensity $I_s = 0.5$ as only light source, so the light position and vector $\vec{L}_s$ equals the eye position and vector. Line illumination with normal vectors $\vec{n}_l(\vec{P})$ is applied as described in Sec. 2.1. Hence, the bidirectional reflectance distribution function (BRDF) is the Phong reflection model.

2.2. Cardiac Blood Flow

Blood flow in the great heart vessels, such as the aorta and pulmonary artery, is typically laminar (straight; follows the vessel's course) with a parabolic velocity profile where the highest velocities are in the center. Aberrant flow patterns (vortex flow) occur only in specific locations. For instance, the formation of a slight helix in the aortic arch is normal. Additional vortices are indicators for various pathologies [HSD13]. Hence, extracting [KGP*13] or highlighting such flow patterns is beneficial for the visual flow assessment performed by medical researchers.

Such qualitative analysis of blood flow patterns is vital for a better understanding of different cardiovascular diseases. For example, a malfunctioning aortic valve with an altered opening characteristic can cause vortex flow formation directly behind the valve [BHB*13]. Vortex flow close to the vessel wall induces high shear forces that promote a widening of the vessel wall (aneurysm development) [BSK*14]. A pathologically widened vessel further increases the probability of emerging vortex flow [MFK*12].

4D PC-MRI Data: Each of the three acquired *phase images* contains flow velocities in one spatial dimension with a spatio-temporal resolution of about $2\times2\times3$ mm /40 ms. They represent a discrete, time-resolved, three-dimensional vector field of one heart beat that was averaged in about 10 min of scanning.

In our framework, the data are pre-processed with single-step 4D phase unwrapping [LSJW15]. A phase wrap is a scan artifact where flow seemingly runs into the opposite direction. A graph cut-based 3D vessel segmentation is performed using the LEMON graph library. This is the basis for a surface extraction via marching cubes that is then smoothed with a volume-preserving λ/μ Laplacian filter [TZG96] with cotangent weights [DMSB99]. For an extensive overview regarding the processing of cardiac 4D PC-MRI data we refer to the survey article by Köhler et al. [KBV*16].

We use a GPU-implemented Runge-Kutta 4 forward and backward integration of 30.000 pathlines per vessel that are randomly seeded (uniform distribution) within the vessel segmentation at random temporal positions. Due to the temporally equidistant step size of heart beat length /200 each line can have a maximum of 200 points. The integration is restricted to the vessel segmentation, so less points are usually integrated before the domain is left (a stop criterion).

In the background we show a direct volume rendering (maximum intensity projection (MIP)) of a high-contrast 3D image

that was derived from the flow data as anatomical context. This temporal MIP contains per voxel the highest velocity during the cardiac cycle. The extracted vessel surface is shown as a silhouette.

3. Requirements

Necessary steps to adapt LineAO to dynamic scenes need to be examined. Blood flow patterns deviating from the expected laminar flow should be the focus of perceptual enhancement. Frame coherence is required to reduce visual clutter during animations. (Semi-)Transparency is problematic in combination with SSAO. Thus, particle trails have to be adapted accordingly. Our targeted user group are medical researchers (radiologists with cardiovascular application and cardiologists), so we attach importance to the ease of use and aim at a solution without any user-specified parameters.

4. Anisotropic LineAO

Like SSAO, LineAO is a real-time capable screen-space method that is calculated in each frame. Thus, the method is already suitable for dynamic scenes from a performance point of view.

4.1. Particle Trail Tapering

Cardiac blood flow animations often display particles with a semi-transparent trail so that the particle position, which matches the current animation time, is fully opaque, and transparency decreases (fading) with increasing temporal distance [VOB*10]. A maximum temporal distance parameter, where opacity becomes 0, regulates the trail length. In addition, the percentage $p \in [0,1]$ denotes where between particle position and trail end the fading starts. Our default value is $p = 1/3$.

Transparency in combination with LineAO is problematic. Since the depth buffer contains only the depth of the fragment closest to the camera while colors are derived from back-to-front blending, AO calculation does not produce the desired results. A corresponding adaption is non-trivial and might increase the computational effort tremendously. Therefore, we omit the use of (semi-)transparent particle trails and employ line tapering ("*gradually narrowing line ends*") instead, as described by Everts et al. [EBRI09]. This has a similar fading effect while operating at full opacity. The percentage p now specifies where the tapering starts. From there, the line width linearly decreases to a minimum of $0.25\times$ the original width (ad-hoc parameter) at the trail ends (see Fig. 2). We show trails in both forward and backward temporal direction.

(a) **(b)**

Figure 2: *Particle trail without (a) and with (b) tapering [EBRI09].*

4.2. Alignment Factor

We aim to improve the quick recognition of major flow structures, such as crossing line bundles. This resembles the goal of illustrative visualizations where simplified depictions are established using feature lines.

With classic LineAO, line bundles with similar flow directions are shown highly detailed (see Fig. 1b). Here, it can be challenging to grasp global flow characteristics at one glance. This problem gets amplified during animations. Therefore, we add a line alignment factor $g_l^{align}(\vec{\omega},\vec{P})$ to the occlusion calculation (Eq. 4) that dampens the AO effect if lines have a similar direction (tangent). Since LineAO operates in screen-space, we decided to determine the alignment there as well with the same sampling scheme that is already used for the depth and lighting terms. The pathline rendering in our framework is realized as line strip with adjacency, i.e., the geometry shader receives four points per line segment where the second and third point are the actual line segment and the first and fourth are predecessor and successor, respectively. First, the points are transformed to screen-space via multiplication with the modelview and projection matrices. Then, the screen-space tangents $\vec{t}_{\text{screen}} \in \mathbb{R}^2$ for point two and three are calculated via central differences and subsequently normalized. We perform no special treatment of invalid tangents for the rare case where two points of a line segment perfectly overlap in screen-space. During the first deferred rendering pass, we store the angle α of each screen-space tangent to the screen's x-axis:

$$\alpha \;=\; |\,\vec{t}_{\text{screen}} \cdot (1,0)^{\mathrm{T}}\,| \;=\; |\,\vec{t}_{\text{screen}}.x\,| \tag{10}$$

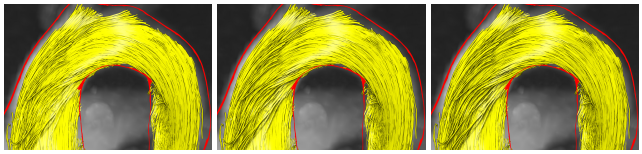
For storage, we use a new angle map (R component only) of which we also require s_r mipmap levels, i.e., the original texture and two further levels of a Gaussian pyramid. The alignment factor is calculated in the fragment shader of the second rendering pass as:

$$g_l^{align}(\vec{\omega},\vec{P}) = \max\Big[A_0,\ \min\Big(1,\ I \cdot h\big(|\alpha(\vec{P}) - \alpha(\vec{P}+\vec{\omega})|\big)\Big)\Big] \tag{11}$$

where $|\alpha(\vec{P}) - \alpha(\vec{P}+\vec{\omega})|$ is the absolute angle difference of the current pixel $\vec{P}$ and the evaluated neighbor sample $\vec{P}+\vec{\omega}$ and $h(x)$ is the Hermite polynomial (Eq. 6) that we reuse for fine-tuning. Factor I controls the intensity of the darkening. We use $I = 15$ (see Fig. 3). $A_0 = 0.1$ (our default) prevents that AO is nullified in case of parallel lines. The alignment factor is then incorporated in Eq. 4:

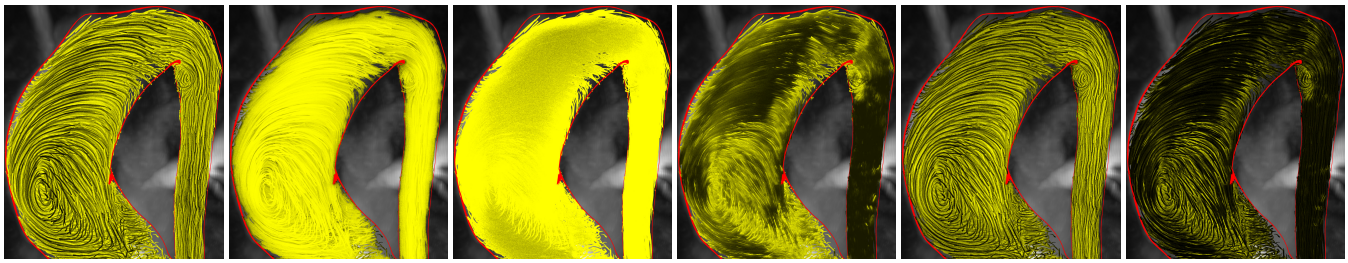
$$g_l(\vec{\omega},\vec{P}) \;=\; g_l^{depth}(\vec{\omega},\vec{P}) \cdot g_l^{light}(\vec{\omega},\vec{P}) \cdot g_l^{align}(\vec{\omega},\vec{P}) \tag{12}$$

Figs. 1 and 8 show a direct comparison of classic and anisotropic LineAO. Fig. 4 depicts the single factors of the AO computation.



(a) $I = 5$ **(b)** $I = 15$ **(c)** $I = 25$

Figure 3: *Intensity scaling I of the alignment factor* g_1^{align} *(Eq. 11). We use* $I = 15$ *(b) as default.*



(a) $\overline{V_l} = 1 - V_l$ **(b)** g_1^{depth} **(c)** g_1^{light} **(d)** g_1^{align} **(e)** $\overline{V_l} \cdot g_1^{\text{depth}} \cdot g_1^{\text{light}}$ **(f)** $\overline{V_l} \cdot g_1^{\text{depth}} \cdot g_1^{\text{light}} \cdot g_1^{\text{align}}$

Figure 4: *The visibility factor V_l (a, Eq. 3), the depth factor g_1^{depth} (b, Eq. 5), the light factor g_1^{light} (c, Eq. 8), and the alignment factor g_1^{align} (d, Eq. 11) of the AO calculation (Eq. 12) as well as their combinations used in classic (e) and anisotropic LineAO (f).*

4.3. Depth-Dependent Halos

Halos are suitable for combination with LineAO. However, regions with similar flow directions appear not as homogeneously as we desire with uniform halos. As a solution, we employ depth-dependent halos [EBRI09]. They are not suited for semi-transparent lines, but this is not a problem, as we are bound to full opacity due to LineAO. We use $d_{\max} = 0.01$ for the halos' maximum depth offset, as suggested by the authors. Moreover, we chose $f_{\text{displacement}}(x) = \sqrt{x}$, because "*using x^2 results in less emphasis for the bundles but more individual lines being visible, while $\sqrt{x}$ has the opposite effect*", according to the authors. Fig. 5 shows a comparison between different halo types.

4.4. Implementation Details

We perform no special handling of over- or underestimation of AO values at the viewport boundaries, as suggested by Eichelbaum et al. [EHS13]. However, this is also not performed in their open source implementation in OpenWalnut (see file "*WGEPostprocessor-fragment.glsl*"). Hermite interpolation (Eq. 6) is provided by GLSL's smoothstep function.

Line Illumination: Against the recommendation, we do also use the ambient and diffuse terms of the Phong illumination instead of only additively combining the specular term. We found many situations where this was beneficial for the spatial perception. Fig. 6 shows an example.

Zoom Map: LineAO requires the size of a unit sphere at world-space position $\vec{p} \in \mathbb{R}^3$ after the screen-space projection was performed. We render pathlines as view-aligned quads. This includes the calculation of a world-space vector $\vec{o}$ that is orthogonal to the current viewing direction $\vec{e}$ and the line tangent $\vec{t}$:

$$\vec{o} = \frac{\vec{e} \times \vec{t}}{\| \vec{e} \times \vec{t} \|} \tag{13}$$

$$\text{with} \quad \vec{e} = \vec{p} - \frac{\vec{e'}.xyz}{\vec{e'}.w} \quad \text{and} \quad \vec{e'} = M^{-1} \cdot (0,0,0,1)^{\mathrm{T}}$$

$M^{-1} \in \mathbb{R}^{4\times4}$ is the inverse of the current modelview matrix. The zoom value z is then obtained as:

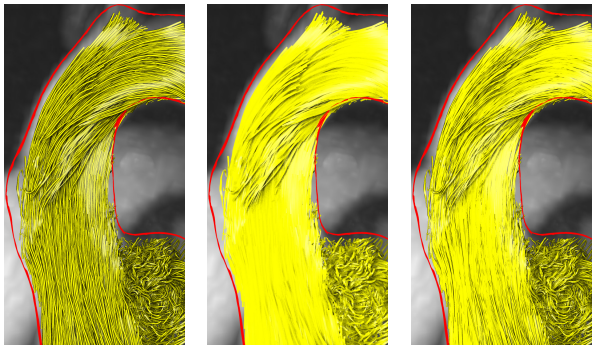
$$z = \| \frac{\vec{p'_0}.xyz}{\vec{p'_0}.w} - \frac{\vec{p'_1}.xyz}{\vec{p'_1}.w} \| \tag{14}$$

$$\text{with} \quad \vec{p'_0} = P \cdot M \cdot (\vec{p},\, 1)^{\mathrm{T}} \quad \text{and} \quad \vec{p'_1} = P \cdot M \cdot (\vec{p} + \vec{o},\, 1)^{\mathrm{T}}$$

$P \in \mathbb{R}^{4\times4}$ is the current projection matrix. It is important to calculate z for each point of the line segment in the geometry shader separately. The G-buffer creation in general should be done with disabled *GL_BLEND* and *GL_ALPHA_TEST*.

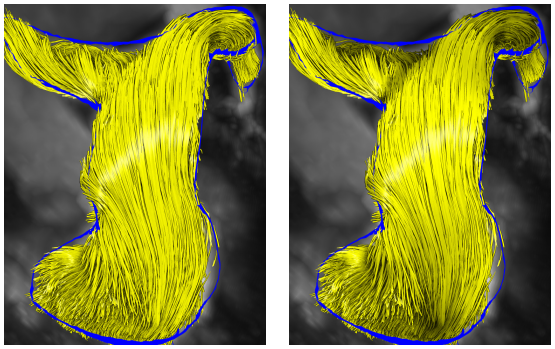
Depth Linearization: Linearization of the depth value $d = gl\_FragCoord.z$ is performed in the fragment shader as:

$$d_l = \frac{z_{\text{near}} \cdot z_{\text{far}}}{z_{\text{far}} - d \cdot (z_{\text{near}} + z_{\text{far}})} \tag{15}$$



(a) **(b)** **(c)**

Figure 5: *Anisotropic LineAO with uniform (a), no (b), and depth-dependent halos (c).*



(a) **(b)**

Figure 6: *Anisotropic LineAO and Phong shading without (a) and with (b) the ambient and diffuse terms.*

Noise in the AO Texture: LineAO employs a 2D noise texture for random sample $\vec{\omega}$ generation. We choose 1024×1024 as texture size, since too small values led to block artifacts in the AO map. The random vectors $\in \mathbb{R}^3$ are already normalized on the CPU.

As typical for SSAO, the AO texture is grainy. Smoothing can be used as a post-processing step. We tried a separated binomial filter of kernel size 3 with up to 10 iterations. The results were acceptable for larger darkened regions. However, the blurring of fine shadows that then leaked into actually unshadowed regions was misleading. Thus, we do not recommend such a smoothing step.

Further Parameter Choices: We use the suggested default parameters $s_r = 3$, $s_h = 32$, and $r_0 = 1.5 \cdot z(\vec{P}) \cdot$ line width. The 4D PC-MRI datasets are scaled in millimeters and our default line width is 0.5 mm. 10 % on each side of the line are colored black, which yields uniform halos. For depth-dependent halos, we double this percentage (experimentally determined). $+1$ on the one side and -1 on the other side of a pathline's view-aligned quad are passed from the geometry to the fragment shader. Black is applied if the absolute of the interpolated value is larger than 0.8.

Separate Rendering of Multiple Vessels: Our rendering pipeline processes each vessel as individual object. Thus, each vessel performs its own deferred shading. To ensure correct depth testing, we use *glBlitFrameBuffer* to copy the current main depth buffer to the G-buffer, then perform the anisotropic LineAO as described, and finally copy the depth buffer back from the G-buffer to the main depth buffer.

5. Results and Discussion

We performed a qualitative evaluation in collaboration with our clinical partners, who are also co-authors of this paper. We start by describing various datasets of patients with different pathologies and healthy volunteers and then proceed with a discussion and performance analysis. All images are screenshots of the animations during systole, when the blood is pumped.

5.1. Case Descriptions

Figs. 8a–c show a patient with an inherited aortic valve malformation named bicuspid aortic valve (BAV). Here, the aortic valve consists of two instead of three leaflets, which causes altered opening characteristics. As a result, vortex flow is often observed behind the valve. Figs. 8d–f depict a patient with a bypass – an artificial course to circumvent a blocked vessel (the vessel section from top right to middle right in the images). In this case, there was a severe narrowing in the aortic arch. The flow is spiraling between the aortic valve and the beginning of the bypass. The patient in Figs. 8g–i has an aneurysmatic ascending aorta (pathologic widening) that causes prominent vortex flow in this vessel section. Figs. 8j–l show a healthy volunteer with mainly laminar flow. The patient in Figs. 8m–o has an aneurysm (top right in the images) with a huge vortex in it that is present during the whole cardiac cycle. Fast flow passes the aortic arch, impinges on the wall on the right, and then starts swirling upwards and downwards. Figs. 8p–r are from a dataset of a pig with a BAV. There is severe vortex flow in nearly the whole aorta. Fig. 1 shows the descending aorta of a patient with an aortic arch prosthesis with pronounced vortex flow in this vessel section. The healthy volunteer in Figs. 3 and 5 has physiological, slight, helical flow in the aortic arch. Fig. 6 is the pulmonary artery of a patient with pulmonary valve defect. The valve does not close properly during diastole (when the ventricles are refilled) and allows a percentage of blood to flow back from the pulmonary artery into the right ventricle. Fig. 7 is the normally functioning aorta of the same patient.

5.2. Discussion

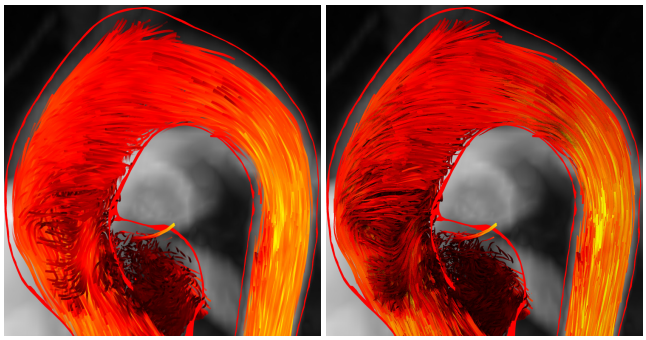
Flow with similar directions appears homogeneous while complex flow is properly highlighted by our anisotropic LineAO. More precisely, vortex boundaries are enhanced, the look into vortex cores, spiral patterns, and complex flow in the heart chambers. A side effect is that also the majority of diastolic flow is darkened due to slow, more or less random flow directions. Our clinical collaborators appreciate the quicker recognition of seemingly interesting flow with our technique. Though, when they want to evaluate individual flow courses, they prefer classic LineAO. A user study will be necessary to verify the benefit of our method.

The render window's background color may play a role in the perception as well. If there are gaps between sparse lines, this can be misleadingly interpreted as shadowing if the background is dark. Though, this does not occur for dense lines that we have in the majority of the cases. Also, our background (DVR MIP of the dataset) is usually in a light gray, showing the vessels.

In the cardiac context, lines are often colored according to their velocity. Fig. 7 shows a comparison of this approach with and without anisotropic LineAO. The colors are strongly changed due to the darkening. The clinicians found this misleading during qualitative quantification. Thus, we do not recommend color-coding in combination with anisotropic or classic LineAO.

5.3. Performance

Our test computer has a GeForce GTX 980. The size of our render window was 853×917 pixels and we use a $2\times$ supersampling



(a) **(b)**

Figure 7: *Color-coded velocities in a heat scale without (a) and with (b) anisotropic LineAO. Line illumination and halos are not used. We do not recommend the use of anisotropic as well as classic LineAO for qualitative quantification due to the altered colors.*

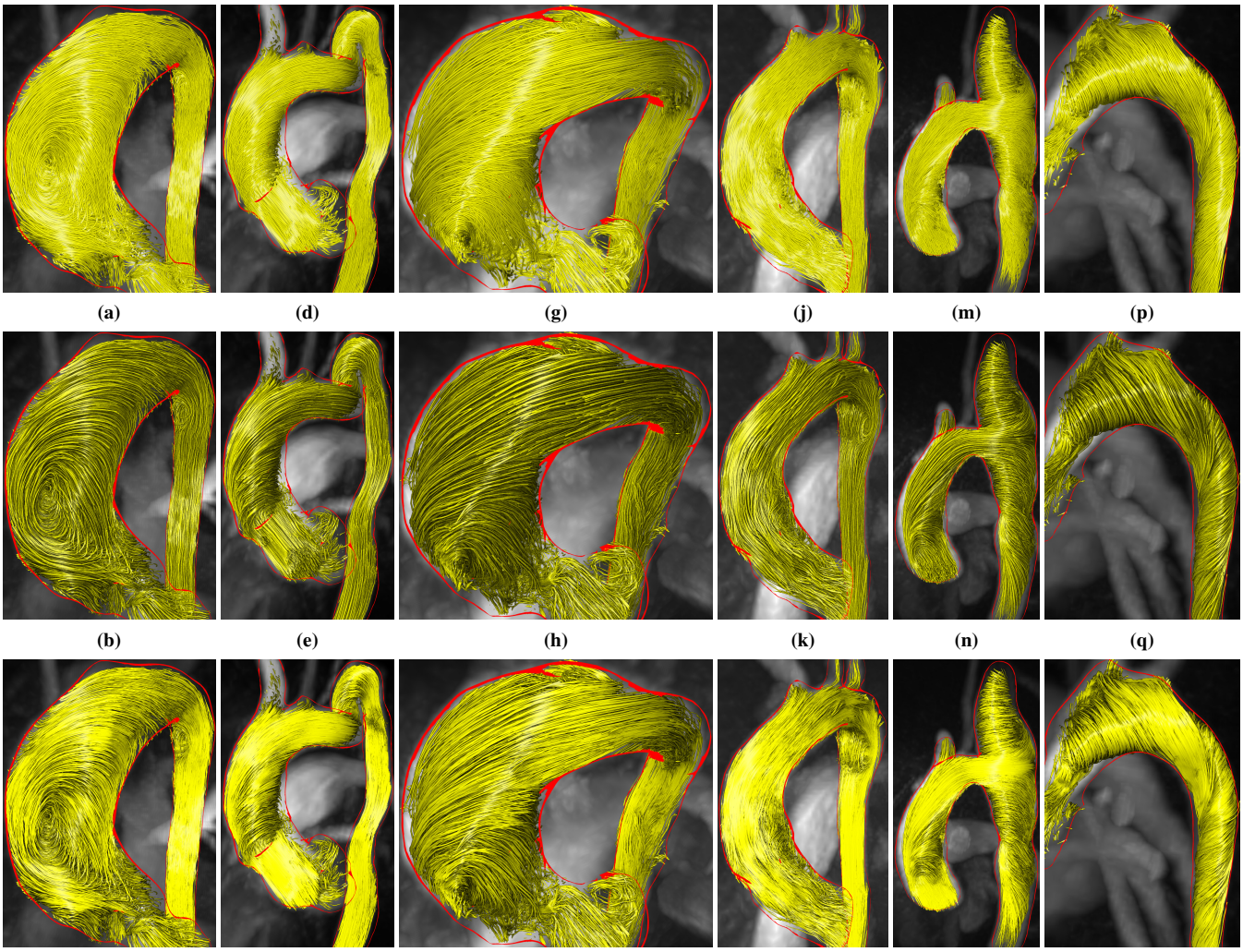


Figure 8: *Full flow visualizations during systole (when the blood is pumped) of different patients. Top row: Line illumination and uniform halos as reference. Middle row: Ambient occlusion for lines (LineAO) [EHS13]. Bottom row: Our proposed anisotropic LineAO with depth-dependent halos [EBRI09].*

Animation	Halo	LineAO	FPS
off	uniform	classic	64
off	uniform	anisotropic	63
off	depth-dependent	classic	60
off	**depth-dependent**	**anisotropic**	**60**
on	uniform	classic	175 / 205
on	uniform	anisotropic	170 / 195
on	depth-dependent	classic	175 / 200
on	**depth-dependent**	**anisotropic**	**165 / 190**

Table 1: *Performance with different configurations.*

anti-aliasing (SSAA), i.e., twice the size per dimension (1706 × 1834 pixels) is rendered and then downsampled to the original size. 30.000 pathlines were integrated. We used this amount for all screenshots in this paper. Table 1 shows the achieved FPS with different configurations. Classic and anisotropic LineAO have a comparable performance. Depth-dependent halos had a rather small influence on the FPS. During animations the FPS depend on the number of drawn line segments. The first value represents a time point during systole, the second one during diastole. Consequently, the particle trail length has an influence as well during animation. We used 25 ms in both forward and backward direction for this test in a dataset with 833 ms heart beat length.

6. Conclusion and Future Work

We presented anisotropic LineAO as an adaption of ambient occlusion for lines (LineAO) [EHS13] that was introduced as a screen-space AO variant for line bundles. We discussed the adaption to dynamic data and necessary steps to make the method feasible for the cardiac blood flow context. To reduce visual clutter we incorporated a line alignment factor in the

AO calculation that diminishes darkening if lines' screen-space tangents are locally similar. To enhance the homogeneous display of similar flow directions, we incorporated depth-dependent halos [EBRI09] instead of uniform ones. This combination allows to show more flow context while still being able to quickly recognize prominent flow structures even without filtering vortex flow. A performance analysis has shown the real-time capability of our method. Anisotropic LineAO seems to be suitable for flow pattern analysis, however, a perceptual user study needs to be conducted in a future work to verify this claim. Our method should not be used for color-coded qualitative quantification, as colors are changed.

The ambient occlusion calculation could be further extended by incorporating velocity magnitudes or other attributes. However, an advantage of the used angles between screen-space tangents is the fix value domain across arbitrary datasets. Hence, our method is directly applicable to other contexts, such as meteorologic data.

We require no user-adjusted settings, since reasonable defaults are provided for all parameters. Due to the feedback by our clinical collaborators, we provide anisotropic LineAO for the enhanced flow structure recognition in our software, but also classic LineAO for the evaluation of individual flow courses.

Acknowledgments: This work was funded by the German Research Foundation (PR 660/18-1). The authors thank B. Behrendt and H. Seibt for fruitful discussions.

References

[BBD*09] Boucheny C., Bonneau G.-P., Droulez J., Thibault G., Ploix S.: A perceptive evaluation of volume rendering techniques. *ACM Trans Appl Percept 5*, 4 (2009), 23:1–24. 2

[BHB*13] Bissell M. M., Hess A. T., Biasiolli L., Glaze S. J., Loudon M., Pitcher A., Davis A., Prendergast B., Markl M., Barker A. J., Neubauer S., Myerson S. G.: Aortic dilation in bicuspid aortic valve disease: Flow pattern is a major contributor and differs with valve fusion type. *Circ: Cardiovasc Imag 6*, 4 (2013), 499–507. 3

[BMGS13] Born S., Markl M., Gutberlet M., Scheuermann G.: Illustrative visualization of cardiac and aortic blood flow from 4D MRI Data. In *Proc: PacificVis* (2013), pp. 129–36. 2

[BSK*14] Burris N. S., Sigovan M., Knauer H. A., Tseng E. E., Saloner D., Hope M. D.: Systolic flow displacement correlates with future ascending aortic growth in patients with bicuspid aortic valves undergoing magnetic resonance surveillance. *Investig Radiol 49*, 10 (2014), 635–9. 3

[DMSB99] Desbrun M., Meyer M., Schröder P., Barr A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proc: SIGGRAPH* (1999), pp. 317–24. 3

[DP80] Dormand J. R., Prince P. J.: A family of embedded Runge-Kutta formulae. *J Comput Appl Math 6*, 1 (1980), 19–26. 2

[EBRI09] Everts M. H., Bekker H., Roerdink J. B. T. M., Isenberg T.: Depth-dependent halos: Illustrative rendering of dense line data. *IEEE TVCG 15*, 6 (2009), 1299–306. 2, 4, 5, 7, 8

[EHS13] Eichelbaum S., Hlawitschka M., Scheuermann G.: LineAO – Improved three-dimensional line rendering. *IEEE TVCG 19*, 3 (2013), 433–45. 1, 2, 5, 7

[GRT14] Günther T., Rössl C., Theisel H.: Hierarchical opacity optimization for sets of 3D line fields. *CGF 33*, 2 (2014), 507–16. 2

[GTG17] Günther T., Theisel H., Gross M.: Decoupled opacity optimization for points, lines and surfaces. *CGF 36*, 2 (2017), 153–62. 2

[HSD13] Hope M. D., Sedlic T., Dyverfeldt P.: Cardiothoracic magnetic resonance flow imaging. *J Thorac Imag 28*, 4 (2013), 217–30. 3

[IG97] Interrante V., Grosch C.: Strategies for effectively visualizing 3D flow with volume LIC. In *Proc: IEEE Vis* (1997), pp. 421–4. 2

[KBV*16] Köhler B., Born S., Van Pelt R. F. P., Hennemuth A., Preim U., Preim B.: A survey of cardiac 4D PC-MRI data processing. *CGF* (2016), Epub ahead of print. 3

[KGP*13] Köhler B., Gasteiger R., Preim U., Theisel H., Gutberlet M., Preim B.: Semi-automatic vortex extraction in 4D PC-MRI cardiac blood flow data using line predicates. *IEEE TVCG 19*, 12 (2013), 2773–82. 2, 3

[Lan02] Landis H.: Production-ready global illumination. In *ACM SIGGRAPH Course Note #16: RenderMan in Production* (2002), pp. 87–102. 2

[LB00] Langer M., Bülthoff H. H.: Depth discrimination from shading under diffuse lighting. *Perception 29*, 6 (2000), 649–60. 2

[LSJW15] Loecher M., Schrauben E., Johnson K. M., Wieben O.: Phase unwrapping in 4D MR flow with a 4D single-step Laplacian algorithm. *JMRI 43*, 4 (2015), 833–42. 3

[MFK*12] Markl M., Frydrychowicz A., Kozerke S., Hope M. D., Wieben O.: 4D flow MRI. *JMRI 36*, 5 (2012), 1015–36. 3

[Mit07] Mittring M.: Finding next gen: CryEngine 2. In *ACM SIGGRAPH 2007 Courses* (2007), pp. 97–121. 2

[MKP*16] Meuschke M., Köhler B., Preim U., Preim B., Lawonn K.: Semi-automatic vortex flow classification in 4D PC-MRI data of the aorta. *CGF 35*, 3 (2016), 351–60. 2

[MPSS05] Mallo O., Peikert R., Sigg C., Sadlo F.: Illuminated lines revisited. In *Proc: IEEE Vis* (2005), pp. 19–26. 3

[PBC*16] Preim B., Baer A., Cunningham D., Isenberg T., Ropinski T.: A survey of perceptually motivated 3D visualization of medical image data. *CGF 35*, 3 (2016), 501–25. 2

[PVH*03] Post F. H., Vrolijk B., Hauser H., Laramee R. S., Doleisch H.: The state of the art in flow visualisation: Feature extraction and tracking. *CGF 22*, 4 (2003), 775–92. 2

[Ram88] Ramachandran V. S.: Perception of shape from shading. *Nature 331*, 6152 (1988), 163–6. 2

[SAG*14] Stankovic Z., Allen B. D., Garcia J., Jarvis K. B., Markl M.: 4D flow imaging with MRI. *JCDT 4*, 2 (2014), 173–92. 2

[TZG96] Taubin G., Zhang T., Golub G. H.: Optimal surface smoothing as filter design. In *Proc: ECCV*. 1996, pp. 283–92. 3

[VOB*10] Van Pelt R. F. P., Oliván Bescós J., Breeuwer M., Clough R. E., Gröller M. E., ter Haar Romeny B. M., Vilanova A.: Exploration of 4D MRI blood flow using stylistic visualization. *IEEE TVCG 16*, 6 (2010), 1339–47. 4

[Wan92] Wanger L.: The effect of shadow quality on the perception of spatial relationships in computer generated imagery. In *Proc: Symp Interact 3D Graph* (1992), pp. 39–42. 2

[WFG92] Wanger L. C., Ferwerda J. A., Greenberg D. P.: Perceiving spatial relationships in computer-generated images. *IEEE Comput Graph Appl 12*, 3 (1992), 44–51, 54–8. 2

[YHGT10] Yang J. C., Hensley J., Grün H., Thibieroz N.: Real-time concurrent linked list construction on the GPU. In *CGF* (2010), pp. 1297–304. 2

[YT10] Yuksel C., Tariq S.: Advanced techniques in real-time hair rendering and simulation. In *ACM SIGGRAPH 2010 Courses* (2010), pp. 1:1–168. 2

[ZSH96] Zöckler M., Stalling D., Hege H.-C.: Interactive visualization of 3D-vector fields using illuminated stream lines. In *Proc: IEEE Vis* (1996), pp. 107–13. 2, 3