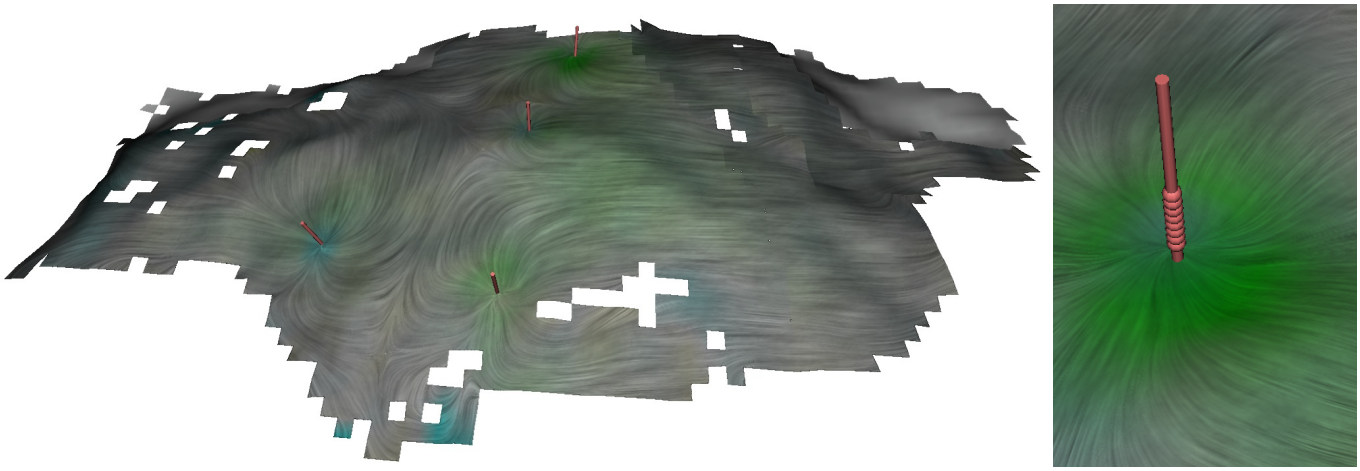
# Visualizing 3D flow of black-oil reservoir models on arbitrary surfaces using projected 2D line integral convolution

Thiago Marques Toledo Waldemar Celes
Tecgraf/PUC-Rio – Computer Science Department
Pontifical Catholic University of Rio de Janeiro, Brazil
Email: {tmt_br, celes}@tecgraf.puc-rio.br



(a) Flow on a layer surface (b) Flow near a producer well

Fig. 1: Visualization of a 3D oil flow of a black-oil reservoir model.

***Abstract*—In the oil industry, clear and unambiguous 3D flow visualization techniques are very important to inspect the results of numerical simulation of black-oil reservoir models. In this paper, we revisit the use of line integral convolution (LIC) for imaging 3D vector fields on arbitrary surfaces and apply it to reservoir visualization. We use a GPU-based method to image the tangential component using the conventional 2D LIC in projected screen space and propose the use of color to encode the normal component. To attach the flow imaging to the 3D surfaces, avoiding image flickering while manipulating the model, we propose a simple scheme based on randomly generated texture coordinate, avoiding the use of a solid 3D texture noise. For animation, we adjust the use of filters to ensure that the animation speed varies in accordance to the field magnitude. We also explore the visualization of multiphase (oil, gas, and water) flow.**

***Keywords*-line integral convolution; 3D flow visualization; black-oil reservoir visualization.**

## I. Introduction

Oil companies extract petroleum from porous rock structures called black-oil fields or reservoirs. The extraction process is done by drilling wells into the reservoir. The oil is brought to the surface by natural underground pressure or by induced pressure (e.g. injecting water). Soil properties and the placement of the wells along the reservoir determine the extraction rate and the lifetime of the oil field.

Planning an oil field exploration consists in assessing sets of wells to maximize oil recovery. Oil companies make use of reservoir numerical simulation to plan oil field exploration. Traditionally, finite difference simulators are employed, representing the model by a discrete three-dimensional mesh of hexahedral cells. The numerical simulator outputs a large set of data representing cell properties along a simulation time line. Among the generated data are 3D vector fields, specially flow of oil, gas, and water, that must be analyzed by means of computer graphics visualization techniques.

However, to achieve clear and unambiguous 3D flow visualization is challenging. Vector field visualization techniques appear in different flavors [1], [2]. Direct flow visualization samples arrow glyphs throughout the domain and allows immediate view of the overall flow, but is hard to be interpreted in 3D. Streamline extraction and rendering falls into the category of geometric flow visualization and faces the challenge of correctly placing seeds in order to avoid missing important fea-

tures of the flow. Texture-based dense visualization techniques combine the previous approaches and use the directional flow information to sample textures, covering the whole model domain. Lastly, feature-based visualization techniques try to extract important features from the data and show only what is interesting to the user, which may vary depending on data and intended application.

In the oil industry, streamlines are traditionally used to reveal flow behavior, but the resulting images show a sparse representation of the flow and are hard to be interpreted without ambiguity in 3D. That is particularly true for reservoir models in which the flow may become quite complex around dense set of wells. We then focus on the use of dense texture-based techniques to visualize flows from a black-oil reservoir simulation. A natural choice would be the use of 3D line integral convolution (3D LIC) [3], [4]. However, once the 3D image is achieved, it must be visualized using volume rendering algorithms, which brings us to the difficult task of choosing an appropriate transfer function to reveal all the important features of the data. Another disadvantage of 3D LIC is related to the 3D image resolution: to achieve good image quality, it requires large memory consumption and high computational time to generate the LIC image.

In this paper, we propose the use of LIC applied on surfaces in 3D [5] for reservoir flow visualization. We believe that using surfaces as the geometric support for visualizing 3D flows in a reservoir model is an adequate strategy. Cells of a black-oil reservoir model lay on a 3D topological grid, being identified by the triple $(i, j, k)$. All cells of a given $k$ constitute a *layer*, which resembles an irregular terrain, with discontinuities; similarly, cells of a given $i$ or $j$ constitute a *section*. Reservoir models are usually inspected using "exploded" views, separating the model into layers, section, or a combination of both [6].

We revisit the use of line integral convolution (LIC) for imaging 3D vector fields on arbitrary surfaces. We present a GPU-based method to image the tangential component using the conventional 2D LIC in projected screen space. However, while performing the LIC in screen space to render 3D objects we need to ensure that the image of the flow is attached to the object instead of the screen; otherwise, the image would change during user interaction (navigation or model manipulation) resulting in annoying image flickering. We solve that by proposing the use of fixed texture coordinates associated to the vertices, randomly generated, and making use of mipmapping for white noise representation. Although simple, this solution is general, avoid the use of a solid 3D texture, and has showed to be quite effective. We also explore colors both to infer a three-dimensional understanding of the 3D flow represented on the surfaces, encoding the normal component, and to allow multi-phase flow visualization in a single image. For animation, we adjust the use of filters to ensure in-phase motion to represent flow direction, varying animation speed in accordance to the field magnitude.

The rest of this paper is organized as follows. Section II reviews dense flow visualization techniques, especially on 3D surfaces. Section III presents our approach to apply LIC on surfaces. Section IV discusses different uses of color and animation to effectively reveal 3D information. The implementation of our algorithm is presented in Section V. Finally, Section VI analyses achieved results and Section VII draws concluding remarks.

## II. Related Work

Cabral and Leedom [3] introduced line integral convolution (LIC) for imaging two-dimensional vector fields. The basic idea is to convolute a white noise image along streamlines computed from the vector field. For a dense regular representation of a 2D vector field, the algorithm uses two aligned images as input: an image representing the vector field, named *velocity image*, and the white noise. For each pixel, a streamline is computed in the negative and positive directions of the vector field. Then, the white noise is convoluted considering all the pixels traversed by the computed streamline. As a result, streamlines covering the entire domain of the vector field are revealed in the resulting image. This works because neighboring pixels along a same streamline will likely have similar streamlines themselves, except for the extremes, and, therefore, will output pixels with similar resulting luminance. On the other hand, neighboring pixels that belongs to different (in general, parallel) streamlines, consider a completely different set of pixels for the convolution, producing different luminance. As a result, distinguishable streamlines are revealed in the final image. Cabral and Leedom [3] also showed that an extension to 3D vector visualization was straightforward, which was later explored by Interrante and Grosch [4].

The computational cost of LIC is proportional to the length of the streamlines, and relatively long streamlines are needed to achieve good image quality. For that reason, following researches tried to improve the performance of the original LIC algorithm ([7], [8], [9], [10]). Later, van Wijk [11] proposed a different technique for two-dimensional flow visualization, named Image Based Flow Visualization (IBFV), which presented much better performance if compared to the traditional LIC. Today, the computational cost of the original LIC algorithm is alleviated by the current graphics card programming capabilities [12]. Recently, Hlawatsch et al. [13] have proposed the use of hierarchical line integration to speed up LIC computation; their proposal was also designed to take advantage of modern GPUs.

Different techniques were also presented for visualizing flow on 3D surfaces [5]. The achieved results resemble the visualization of a wind tunnel test using surface oil flow [14]. Both Teitzel et al. [15] and Battke et al. [16] extended LIC to visualize flow on arbitrary surfaces tessellated in triangles, performing convolution in object space. Later, van Wijk [17] and also Laramee et al. [18] proposed to apply the IBFV algorithm to 3D surfaces realizing texture advection in image space. As a result, their techniques are fast and independent of the surface mesh's complexity. Weiskopf and Ertl [19] also proposed a GPU-based LIC algorithm for imaging vector fields on surfaces; they perform the convolution in screen space

and use a solid 3D texture noise to ensure frame-to-frame coherency.

We have also used the screen space to perform image convolution using a GPU-accelerated LIC algorithm. Different from previous works, we propose the use of color to infer a three-dimensional understanding of the 3D flow and avoid the use of 3D textures while ensuring frame-to-frame coherency.

## III. Projected LIC

Reservoir simulators output, among other data, vector fields associated to the cells of the model, at different time steps of the simulation. Due to the large amount of data generated, reservoir engineers, in general, opt for outputting only a few critical time steps. This sparse output disallows accurate vector field interpolation, forcing us to visualize each outputted time step as representing steady-state flows.

In a pre-processing phase, we compute vector fields at vertices averaging data associated to neighboring cells. We then send the 3D vector field to the graphics pipeline as vertex attribute. The goal is to visualize these 3D data using surfaces as the geometric support. Layer and section surfaces are built by extracting, for each hexahedral cell, the corresponding central quadrilateral. The vector field is often not aligned with the surfaces. As pointed out by van Wijk [17], visualization of non surface-aligned flow is more difficult to be interpreted. To overcome this difficulty, we propose to apply LIC to the tangential component of the flow and to use color (see Section IV) to depict the normal component.

As illustrated in Figure 2, given the vertex unit normal ($\hat{n}$), we compute the tangential component of the 3D vector field ($\vec{f}$) associated to the corresponding vertex:

$$\vec{f_t} = \vec{f} - (\vec{f} \cdot \hat{n})\hat{n} \tag{1}$$

Because we apply LIC in screen space, we need to project the resulted tangential vector field. Being $\mathbf{v}$ the vertex position in object space, we perform the projection by applying the modelview-projection matrix ($\mathbf{M}$). Denoting projected quantities with apostrophes, we have:

$$\vec{f_t'} = \mathbf{M}(\mathbf{v} + \hat{f}_t) - \mathbf{M}\mathbf{v} \tag{2}$$

In this formula, note that we use the unit tangential vector ($\hat{f}_t$) to ensure that the position $\mathbf{v} + \hat{f}_t$ does not fall behind the camera. This is valid because LIC does not consider vector field magnitude.

We then adjust the resulting projected vector by the screen aspect ratio ($\vec{f_t'} \leftarrow \vec{f_t'}\ w/h$), and apply conventional 2D LIC in screen space. A $z$-value buffer is used to detect depth discontinuity along the streamline to avoid wrong convolution of pixels that are far apart in object space, as proposed by Laramee et al. [18]. The depth value is also used to identify fragments that belong to the background.

### A. Noise mapping

For surfaces, when convolution is done in object space, there is the need to use a solid 3D white noise, which requires a significant amount of memory and limits the level of magnification achievable without blurring the image. In image space, one can consider that both vector field and white noise images are aligned. This allows a direct correspondence between pixels of both images to perform the convolution. Magnification is unlimited, always producing high quality images.

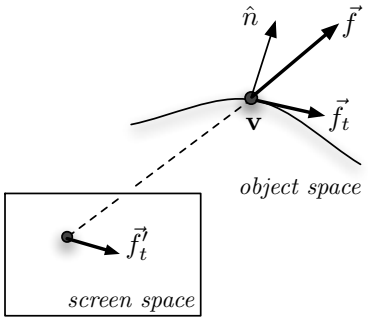



Fig. 2: Projection of tangential vector field.

However, for image-space convolution, if we consider user interaction for manipulating the model or moving the camera, a given part of the model will be projected on different regions of the screen, resulting in annoying image flickering. In order to eliminate this effect, a given triangle of the surface mesh has to be mapped, always, to the same texture region. One can achieve this by employing a solid 3D texture, mapping the white noise in object space, while performing convolution in screen space. To avoid the problems related to using 3D textures, we propose a new strategy to assign texture coordinates to the vertices of the mesh using a 2D noise image, while eliminating image flickering.

In a pre-processing phase, we assign a randomly generated texture coordinate to each vertex of the model. As a result, triangles of the mesh will be mapped into arbitrary regions in texture space. The probability to end up with degenerated texture mapping is too small, and we assume it does not happen. After projection, triangles are mapped to screen, calling for texture magnification or minification. We try to avoid texture magnification through the use a large noise image (in our experiments, a size of 2048 x 2048 has proved to be enough). Texture minification is solved by the use of mipmapping. For each level of the mipmap pyramid, we construct a different white noise image. We then set texture filtering to perform linear interpolation across pyramid levels and to get the nearest value in a given level. In this way, we ensure smooth transition across different levels but preserve aleatory distribution of fetched samples.

## IV. Coloring and animation

The LIC reveals only local vector field tangent, but neither magnitude nor direction are depicted (see Figure 3a). Cabral and Leedom [3] proposed the use of a color scale to map field magnitude, and the use of periodic motion filters to reveal flow direction via animation.

In our case, we can also map the field magnitude as illustrated in Figure 3b. However, this does not suffice, be-

cause the image does not reveal information regarding the normal component of the field. This is important for a correct understating of the 3D flow. To solve that, we can represent the normal component using a different color scale. Since the normal component is normalized, it varies from $-1$ to $1$ (negative numbers indicate the flow is entering the surface); we map negative values to blue, zero value to white, and positive values to red, as shown in Figure 3c.

We then propose the use of a two-dimensional color scale, combining both previous scales, as illustrated in Figure 4. This allows us to map both magnitude and normal component simultaneously in a single image. If the field is aligned to the surface, the scale varies from white to green. The blue color is assigned to negative normal component with high magnitude, while red to positive. A cyan tone indicates that the flow is entering the surface with low magnitude, while a yellow tone indicates the flow is going towards the viewer, also with low magnitude. With this coloring scheme, 3D flow is correctly revealed, even with the use of surfaces as geometric support. Figure 3d illustrates the achieved result. We believe that with little practice the resulting image can be easily and correctly interpreted.

For revealing flow direction via animation, Cabral and Leedom [3] suggested the use of a phase shifted Hanning filter windowed by the Hanning function itself. Our goal was to control the animation speed in a way proportional to the field magnitude. That is, in regions with high magnitude values, the flow should move faster. We applied a *speed factor* ($\lambda$), somehow proportional to the field magnitude, to control phase shift, expressing the kernel by:

$$k(w) = \frac{1+\cos(cw)}{2} \times \frac{1+\cos(dw+\lambda\,\beta)}{2} \tag{3}$$

with $\lambda \in [0,1]$. First, for each pixel, we tried to set $\lambda$ directly proportional to the normalized magnitude value ($\bar{m}$):

$$\lambda_i = \bar{m}_i = \frac{m_i}{m_{max}} \tag{4}$$

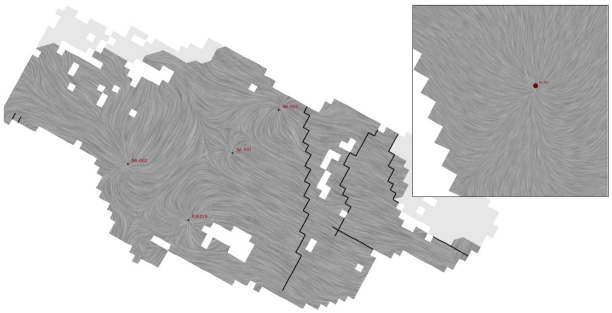
where $m_i$ represents the magnitude field value of a pixel, and $m_{max}$ the maximum magnitude value of the field.

However, with this speed factor, neighboring pixels along a given streamline went out of phase and, after a couple of frames, the animation started moving backwards, especially in areas of high magnitude values. In order to keep neighboring pixels along streamlines in phase while adjusting animation speed, we propose a factor proportional to the quantized normalized magnitude value. In our examples, we have used:

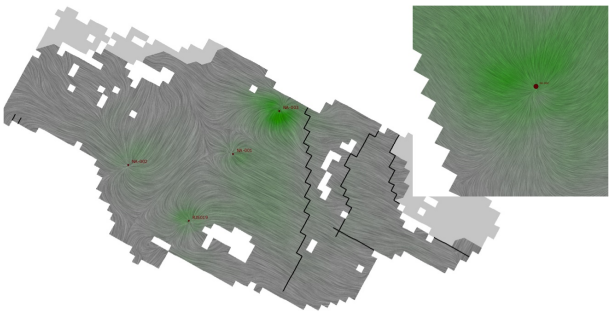
$$\lambda = 0.1 \left\lfloor \frac{\bar{m}_i}{0.1} \right\rfloor \tag{5}$$

In this way, areas of high magnitude values move faster, but all pixels in a same area move in phase at the same pace.

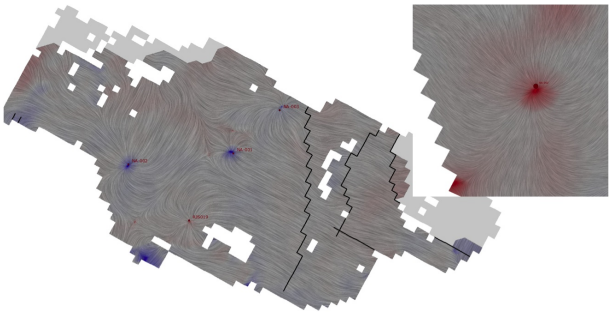
We have also extended our technique to visualize multiphase flow. Reservoir simulation computes flow and saturation change of three phases (oil, water and gas). We then examined the use of colors to simultaneously visualize multiphase flow. In the oil industry, traditionally, visualization of oil, water, and gas data are assigned to green, blue, and red colors, respectively. We than use the three color channels to encode the result of up to three LIC algorithms, one for each phase flow. Figure 5 illustrates the result for a simulation that considers only oil and water. In this image, the color intensity



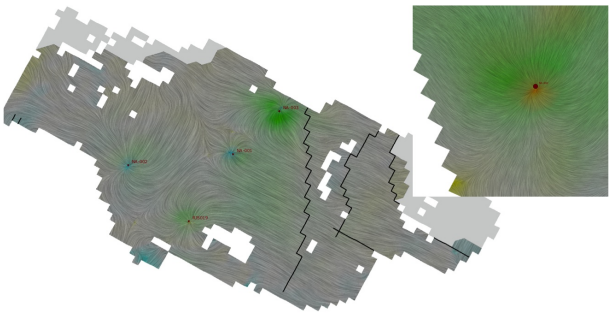
(a) Vector field revealed by LIC



(b) A white-green scale maps magnitude



(c) A blue-red scale maps normal component



(d) A 2D color scale maps both magnitude and normal component

Fig. 3: Proposed coloring scheme.

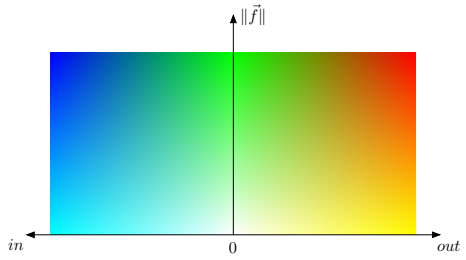



Fig. 4: Proposed two-dimensional color scale.

is given by the saturation value of each phase. The image then depicts both saturation and flow distribution of oil and water throughout the reservoir.

## V. Implementation

The proposed technique is implemented in a two-rendering pass algorithm, coded in GLSL. In the first pass, the 3D surface model is processed and rasterized to multiple rendering targets. The application has to provide the following vertex attributes: position, normal, 3D vector field, and randomly generated texture coordinate. The vertex shader implementation is straightforward and basically transforms the vertex and passes all the vertex attributes (including position) to be interpolated by the rasterizer. The fragment shader applies Equations 1 and 2 to compute the projected tangential vector field. It outputs three images to be used by the second pass:

- *color image*: RGBA image that stores the illuminated color $(r, g, b, \alpha)$ at each pixel; this represents the shaded pixels to be combined to the vector field imaging. In general, this corresponds to a luminance image, but we keep it as a full color image to allow the application to combine textures and transparency to the vector field imaging.
- *texture coordinate image*: RGB image that stores the texture coordinate $(s, t)$ assigned to each pixel. In the third channel, it stores the corresponding pixel depth value $(z)$.

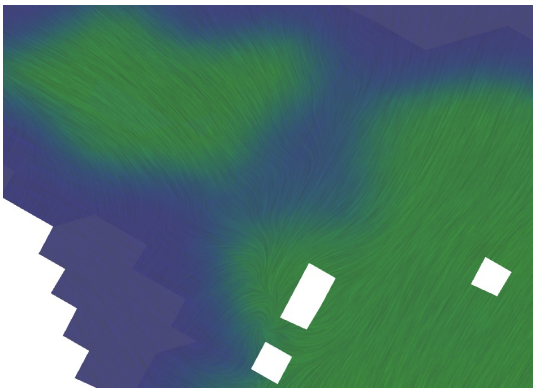


Fig. 5: Multiphase (oil and water) flow visualization. In this case, color intensity is set proportional to saturation.

- *vector field image*: RGBA image that stores the 2D projected tangential vector field $(f'_{t_x}, f'_{t_y})$, using the third channel to store the normal component $(f_n)$, and the forth channel to store the corresponding 3D vector field magnitude $(\|\vec{f}\|)$. The front-facing flag and the interpolated vertex normal is used to appropriately set the normal component sign: negative values for flow entering into the surface and positive values for flow arising from the surface.

The second pass is application independent and operates in screen space. In this pass, the projected vector field image (velocity image) is used to perform conventional 2D LIC in screen space. The normal component and the field magnitude are used to fetch the color palette. The depth value is used to detect pixel belonging to the background and streamline discontinuities. The texture coordinate values are used to fetch the 2D texture noise that is also provided as input. The result of the LIC at each pixel is then combined to the color image provided by the first pass.

## VI. Result Analysis

In this section, we first discuss the benefits of combining projected LIC with an appropriate color encoding for revealing 3D flows using surfaces as the geometric support. The goal of our proposal is to use surfaces as the support for visualizing actual 3D flow, which are, in general, not surface aligned. To be an effective visual representation, the produced images have to clearly and unambiguously reveal the 3D structure of the flow. van Wijk [17] discussed the visualization of non-aligned flow on surfaces. He compared images that depicted the surface of a torus in a vertical flow, as schematized in Figure 6: $\hat{f}_1$ represents the vertical vector field.

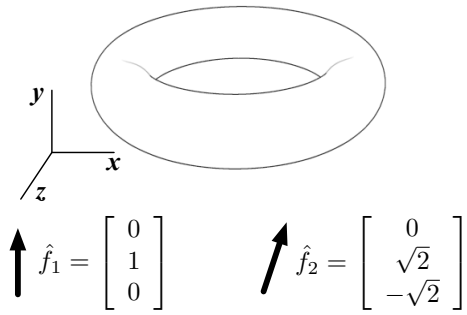
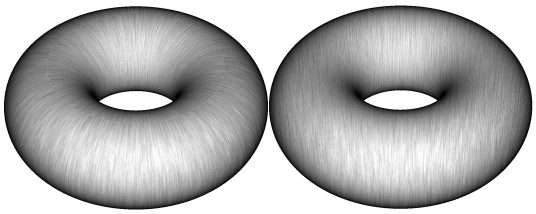


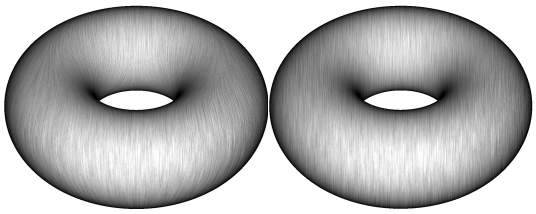

Fig. 6: Surface of a torus in two different flows: $\hat{f}_1$ represents a vertical flow, and $\hat{f}_2$ represents a $45^o$ degree deviated flow.

van Wijk [17] compared the achieved results for two different approaches to deal with non-aligned surface flows. A first image was generated by projecting the flow on the surface and imaging the tangential component, and a second by imaging the 3D flow as it is. We reproduced this experiment with our algorithm and obtained the images shown in Figure 7, which are similar to the ones presented in [17].



(a) Flow projected on surface (b) Flow not projected

Fig. 7: Surface of a torus in a vertical flow.



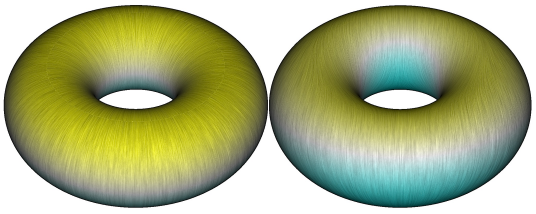
(a) Flow projected on surface (b) Flow not projected

Fig. 8: Surface of a torus in a $45^{o}$-degree deviated flow.

While the first image shows the surface shape clearly, it does not reveal the actual flow. van Wikj [17] concluded that the second image shows the flow field better. In his opinion, the image effectively conveys that this is a simple and uniform flow. In this case, the torus would represent a flow probe for inspecting the involving 3D flow.

We argue that, without an appropriate coloring scheme, both images are unclear and ambiguous. Consider, for instance, the same approaches for a $45^{o}$-degree deviated flow, as illustrated by field $\hat{f}_2$ in Figure 6. Again, the images achieved by projecting the flow on the surface and not projecting are shown in Figure 8. As can be noted, the image obtained by the tangential component reveals a change of the flow, while the image of the 3D flow without projecting on the surface does not reveal the change (compare Figures 7b and 8b).

Additionally, for reservoir flow visualization, the surfaces of layers or sections cannot be viewed as probes, but as actual geometry of the model. It is important to understand the flow in relation to the surfaces. For that reason, we have opted to project the flow on the surface and to use color to depict the normal component. When we apply the proposed two-dimensional color scale, the changes in flow direction are clearly revealed. Figure 9 shows the achieved image assuming a small magnitude value.



(a) A vertical flow (b) A $45^{o}$-degree deviated flow

Fig. 9: Projected flow using the proposed color scheme.

We then test our technique on actual reservoir data. Figure 10 shows the achieved image for visualizing a 3D oil flow on different layers of a simulated reservoir model. The image clearly reveals the flow near a producer well. The green and blue tones indicate high magnitude occurs only near the well. The blue tone on the top layers indicates the flow is going downward. Green tone indicates the precise layer in which the oil is entering the well.

Figure 11 illustrates that the technique can be applied to arbitrary surfaces. In the case, a layer surface is shown together with two crossing sections. Note that different crossing surfaces can be used to help understanding the 3D field based on projected components.

The avoidance of image flickering while interacting with the model and the flow animation speed varying in accordance to the field magnitude can be observed in the video attached as additional material (at JEMS site).

## VII. Conclusion

The numerical simulator of black-oil reservoir models outputs 3D vector fields, specially flow of oil, gas, and water, that must be inspected by reservoir engineers. In this paper, we propose the use of LIC on surfaces to visualize 3D flow of black-oil reservoir models. Layers and sections of reservoir models represent natural geometric supports for inspecting simulation results, and it seems natural to explore these surfaces to reveal flow information as well. As in [17], [18], and [19], we have opted for performing image convolution in screen space, using a GPU-based implementation of the LIC algorithm. We propose the use of a two-dimensional color scale to infer a three-dimensional understanding of the 3D flow represented on the surfaces. We also explore colors for multiphase flow visualization. We present an effective strategy for avoiding image flickering during user interaction, based on randomly generated texture coordinates assigned to vertices and an appropriate scheme for representing and filtering the white noise. We also included a speed factor in the periodic filter proposed by Cabral and Leedom [3] to animate the flow with a speed proportional to the field magnitude.

The proposed technique was validated by reservoir engineers and is currently being integrated into an industrial application. We are currently investigating the use of the proposed technique on terrain surfaces for GIS applications. To improve the achieved images, we plan to test the use of twofold convolution as proposed by Weiskopf [20].

## Acknowledgment

We thank CNPq (Brazilian National Research and Development Council) for the financial support to conduct this research. This work was done in the Tecgraf laboratory at PUC-Rio, which is mainly funded by the Brazilian oil company, Petrobras.

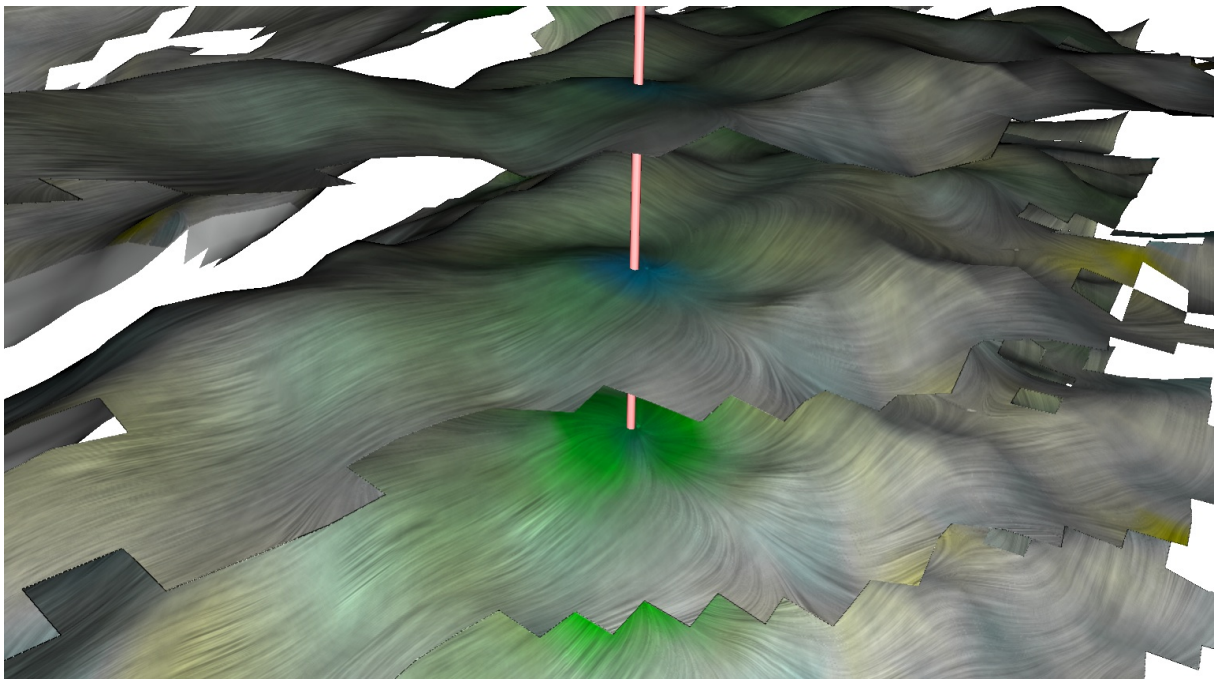


Fig. 10: Visualization of 3D oil flow on different layers (with imposed separation) of a reservoir model. The LIC reveals the tangential component of the flow and colors indicate the normal component and magnitude.

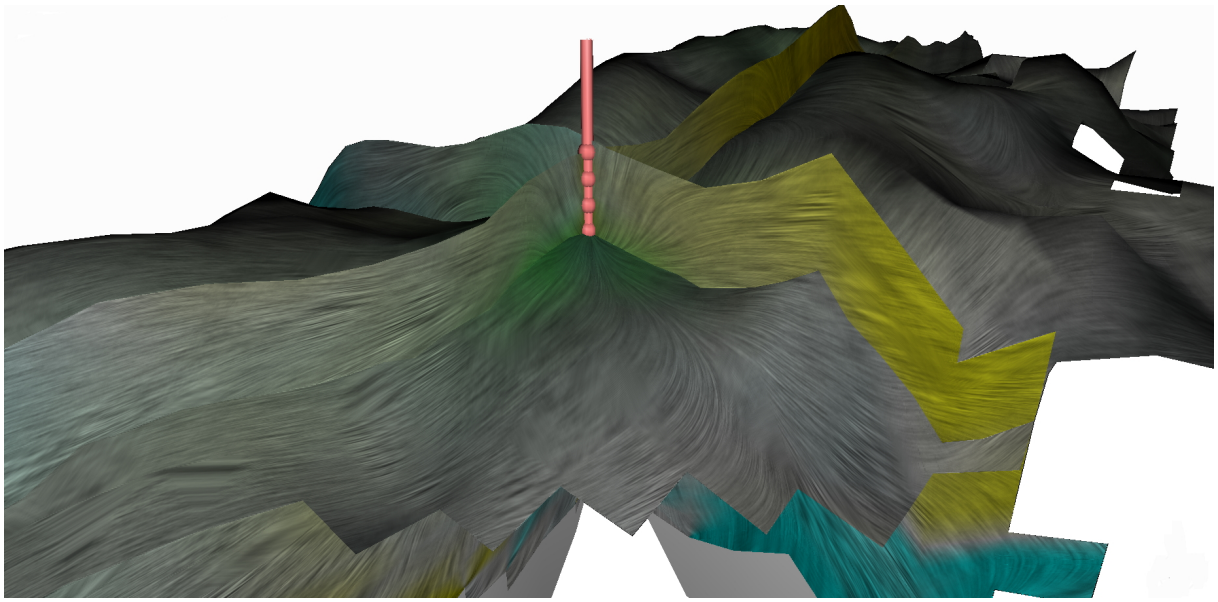


Fig. 11: Visualization of 3D oil flow on different crossing surfaces. In the case, a layer surface is combined with two orthogonal crossing sections.

## REFERENCES


[1] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch, "The state of the art in flow visualisation: Feature extraction and tracking," *Computer Graphics Forum*, vol. 22, no. 4, pp. 775–792, 2003.

[2] R. S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. H. Post, and D. Weiskopf, "The state of the art in flow visualization: Dense and texture-based techniques," *Computer Graphics Forum*, vol. 23, no. 2, pp. 203–221, 2004.

[3] B. Cabral and L. C. Leedom, "Imaging vector fields using line integral convolution," in *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 1993, pp. 263–270.

[4] V. Interrante and C. Grosch, "Strategies for effectively visualizing 3d flow with volume lic," in *Visualization '97., Proceedings*, oct. 1997, pp. 421 –424.

[5] D. Stalling, "Lic on surfaces," in *Texture Synthesis with Line Integral Convolution, ACM SIGGRAPH 97, International Conference on Computer Graphics and Interactive Techniques*, 1997, pp. 51–64.

[6] F. Abraham and W. Celes, "Distributed visualization of complex black oil reservoir models," in *EGPGV– Eurographics Symposium on Parallel Graphics and Visualization*, 2009, pp. 87–94.

[7] D. Stalling and H.-C. Hege, "Fast and resolution independent line integral convolution," in *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '95. ACM, 1995, pp. 249–256.

[8] A. Okada and D. Lane, "Enhanced line integral convolution with flow feature detection," in *In SPIE Vol. 3017 Visual Data Exploration and Analysis IV*, 1997, pp. 206–217.

[9] M. Zockler, D. Stalling, and H.-C. Hege, "Parallel line inntegral convolution," *Parallel Computing*, vol. 23, pp. 975–989, 1997.

[10] H. christian Hege and D. Stalling, "Fast lic with piecewise polynomial filter kernels," in *Mathematical Visualization*. Springer, 1998, pp. 295–314.

[11] J. J. van Wijk, "Image based flow visualization," in *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '02. ACM, 2002, pp. 745–754.

[12] Y. Tan, Y. Shi, K. Tan, B. Qin, Z. Wu, F. Su, and T. Pang, *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2010, vol. 6145, ch. GPU-Based Parallelization Algorithm for 2D Line Integral Convolution, pp. 397–404.

[13] M. Hlawatsch, F. Sadlo, and D. Weiskopf, "Hierarchical line integration," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, pp. 1148–1163, 2011.

[14] N. G. R. Center. (2011, Apr) The beginner's guide to aeronautics. [Online]. Available: www.grc.nasa.gov/WWW/k-12/airplane/

[15] C. Teitzel, R. Grosso, and T. Ertl, "Line integral convolution on triangulated surfaces," in *In WSCG 1997 Conference Proceedings*, 1997, pp. 572–581.

[16] H. Battke, D. Stalling, and H.-C. Hege, *Fast line integral convolution for arbitrary surfaces in 3D*. Springer-Verlag New York, Inc., 1997, pp. 181–ff.

[17] J. J. van Wijk, "Image based flow visualization for curved surfaces," in *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, ser. VIS '03. IEEE Computer Society, 2003, pp. 123–130.

[18] R. S. Laramee, B. Jobard, and H. Hauser, "Image space based visualization of unsteady flow on surfaces," in *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, ser. VIS '03. IEEE Computer Society, 2003, pp. 131–138.

[19] D. Weiskopf and T. Ertl, "A hybrid physical/device-space approach for spatio-temporally coherent interactive texture advection on curved surfaces," in *Proceedings of Graphics Interface 2004*, ser. GI '04, 2004, pp. 263–270.

[20] D. Weiskopf, "Iterative twofold line integral convolution for texture-based vector field visualization," in *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, ser. Mathematics and Visualization. Springer Berlin Heidelberg, 2009, pp. 191–211.