Adding a scalar value to 2D vector field visualization: the BLIC (Bumped LIC)

A. Sanna and B. Montrucchio

Dipartimento di Automatica e Informatica, Politecnico di Torino, Torino, Italy
{sanna,montru}@polito.it

Abstract

Visualization of vector data produced from application areas such as computational fluid dynamics (CFD), environmental sciences, and material engineering is a challenging task. Texture-based methods reveal to be effective, versatile, and suitable for a large spectrum of applications since they allow to obtain high resolution output textures where direction, orientation, and magnitude of the flow can be displayed.
In this paper we present a new method called BLIC (Bumped LIC), which allows both to characterize and visualize interesting structures in the flow and to map an additional scalar value in the output texture by bumps and depressions, leaving colors for further information mapping. Some examples show how the proposed method can enhance the quality of output textures with respect to the classical texture-based approaches.

1. Introduction

Vector field visualization is fundamental for a large spectrum of disciplines where the data obtained by experimental observations and theoretical elaboration need to be graphically displayed. A graphical representation method can be used for two complementary purposes: to visualize carefully the shape and the features of a pattern coherent with the theoretical models, and to infer from a given pattern the types and the distributions of objects actually present, and relevant for the observer.
Actually, most of the graphics icons that can be used to represent a vector use too many pixels; in fact, while a scalar can be visualized varying the color of a pixel, using a segment (or an arrow) to visualize a vector may need much more pixels. In the past years some alternative methods have been suggested, from streamlines (Helman and Hesselink[1]) to particle tracing techniques[2]; the choice of seed points is very important in particle tracing methods to avoid of losing interesting details of the field.
To solve this problem the texture-based methods were introduced; in these techniques all parts of the vector field are represented by using a 2D texture.
The elegance and the effectiveness of these methods were the starting point for new and interesting research in this field. Although several works have been published in the literature[3] [4] [5] [6] [7], some issues must be still tackled. In particular, research is involved in improving the quality of the output textures, in order to allow a better understanding of the vector field under analysis. Moreover, in a large spectrum of applications is required to map in the output texture more information than direction, orientation and magnitude of the flow; for instance, CFD applications may require to map several scalar values like: temperature, pressure, vorticity, and so on. This last problem can be partially tackled using colors by dye advection[8], but multivariate visualization is still an open problem.
The proposed algorithm attempts to combine the texture-based techniques with another well known graphics algorithm called bump mapping[9]. The bump process allows to improve the quality of the output texture providing a way to map an additional scalar value according to a bump texture related to the scalar itself. The scalar value can be mapped leaving colors to map further information, and the bump process can be carried out without delay with standard hardware support. For this reason BLIC allows a real gain in multivariate visualization.
The paper is organized as follows: Section 2 reviews the main texture-based techniques and explains the bump mapping process; the algorithm is presented in Section 3, while

application examples are shown in Section 4. Finally, remarks on the proposed algorithm can be found in Section 5.

2. Previous Work

2.1. Texture based visualization techniques

Texture-based methods attempt to reproduce techniques known from experimental flow visualization such as the observation of randomly dispersed particles or dye injection patterns. The common goal is to produce high resolution images revealing the flow field characteristics: direction, orientation, magnitude, and so on.
Van Wijk[3] proposed to convolve a random (white noise) texture along a straight segment whose orientation is parallel to the direction of the flow. This method (called spot noise) was then extended by bending spot noise, filtering the image to cut low frequency components, and using graphics hardware methods, also on grids with irregular cell sizes (De Leeuw and Van Wijk[10]).
Cabral and Leedom[4] introduced the Line Integral Convolution (LIC) algorithm, which locally filters a white noise input texture along a path of vectors tangent to the field, denoted as streamline.
Given a steady vector field defined by a map $v : \Re^2 \rightarrow \Re^2, x \longmapsto v(x)$, its directional structure can be shown by the integral curves, or streamlines, where an integral curve is a path $\sigma(u)$ having tangent vectors coinciding to the vector field (that is $\frac{d}{du}\sigma(u) = v(\sigma(u))$).
Doing a re-parameterization of $\sigma(u)$ by using arc-length s, we can calculate the line integral convolution (LIC) for a pixel located at $x_0 = \sigma(s_0)$:

$$I(x_0) = \int_{s_0-L}^{s_0+L} k(s-s_0)T(\sigma(s))ds. \tag{1}$$

where $T(x)$ is an input white noise texture, $k(s)$ is the filter kernel (normalized to unity), and the filter length is $2L$.
Being the LIC computationally expensive, Stalling and Hege[5] improved the speed of LIC (fastLIC) more than ten times, by observing that the LIC value computed for one pixel can be re-used, with small modifications, from its neighbor pixels; in this way, the computation is streamline oriented and not pixel oriented as in the conventional LIC.
Zöckler et al.[11] showed a parallel implementation of fastLIC which is able to run in real-time on particular parallel architectures.
Wegenkittl et al.[6] introduced OLIC (Oriented Line Integral Convolution) and then Wegenkittl and Gröller[7] FROLIC (Fast Rendering OLIC). OLIC simulates the use of drops of ink smeared to the underlying vector field. The algorithm can be made faster by positioning small and overlapping disks (FROLIC) in order to simulate the convolution. Besides direction, the length of the pixel traces shows vector orientation and local magnitude of the field. However, OLIC and FROLIC employ sparse textures, therefore, small details of the field may be lost in the visualization.

2.2. Bump mapping

The vector field visualization algorithm proposed in this paper is related to a computer graphics technique called bump mapping. In 1978, James Blinn[9] presented a method simulating the bumps or wrinkles in a surface without the need for geometric modifications to the model. The surface normal of a given surface is perturbed according to a bump map, and the perturbed normal is used instead of the original one when shadows are computed using the Lambertian technique; this approach provides the appearance of bumps and depressions in the surface. Given a point on a surface parameterized by the function $\mathbf{O}(u,v)$, the normal $\mathbf{n}$ at that point is computed by:

$$\mathbf{n} = \mathbf{Q}_u \otimes \mathbf{Q}_v \tag{2}$$

where $\mathbf{Q}_u$ and $\mathbf{Q}_v$ are the partial derivatives in the parameter directions u, and v, and $\otimes$ denotes the outer product. A new displaced point can be defined by adding some amount along the normal at that point:

$$\mathbf{Q}^{'}(u,v) = \mathbf{Q}(u,v) + P(u,v)\frac{\mathbf{n}}{|\mathbf{n}|} \tag{3}$$

where $P(u,v)$ is a perturbation function. The new perturbed normal can be computed as:

$$\mathbf{n}^{'} = \mathbf{Q}_u^{'} \otimes \mathbf{Q}_v^{'} \tag{4}$$

Under the assumption of P small, $\mathbf{n}^{'}$ can be reduced to:

$$\mathbf{n}^{'} = \mathbf{n} + \frac{P_u(\mathbf{n} \otimes \mathbf{Q}_v)}{|\mathbf{n}|} + \frac{P_v(\mathbf{Q}_u \otimes \mathbf{n})}{|\mathbf{n}|} \tag{5}$$

The value of the new (perturbed) normal is based both on the original normal and on the perturbation function, which can be defined mathematically or by a two dimensional lookup table.

3. The BLIC algorithm

The first and main goal of all visualization algorithms is to enhance the understanding of the data to be displayed. In particular, for the flow field visualization, it is worth to allow the user to detect interesting structures such as vortices. The best texture-based techniques can effectively show direction, orientation and magnitude of the flow, and in some cases, using colors, can map one additional scalar value in the resulting texture; unfortunately, some applications require to map and visualize more scalar values.
The proposed work aims to extend the ability of the texture-based techniques known in the literature to map an additional scalar value on the output texture.
The basic idea of this work is to better characterize the flow field structures and to map the information of another scalar value on the texture by a post-processing phase where the bump mapping algorithm is applied on a texture achieved by a classical texture-based technique such as LIC (even if all known non-sparse texture-based methods could be used). In

order to do this, the scalar value to be mapped in the texture can be used to obtain a bump map (also called altitude map). For instance, areas of the flow field characterized by high values of vorticity will be displayed by a bump or a depression according to the sign of the vorticity itself. In this way, interesting areas are better visualized in the texture and an additional scalar value can be mapped. Let us consider as example the vector field visualized by the LIC texture of Figure 1; that field presents six vortical structures with different trend (not visualizable by the LIC algorithm). A bump texture for this example has been computed using Matlab in order to show the different trend of vortices (see Figure 2). The resulting BLIC texture can be found in Figure 3, where it can be noticed as some vortices appear bumped and other depressed in the surface according their trend; therefore, a better characterization of the vortical structures has been obtained and an addition scalar value (the trend of the vortices) has been mapped.

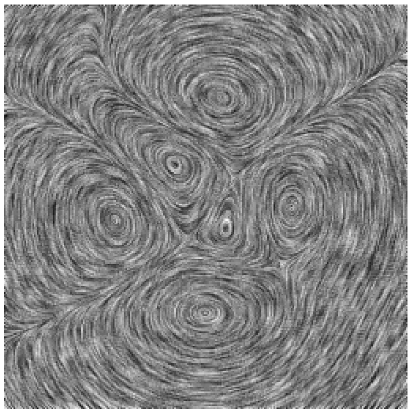
Figure 1: *LIC texture of the example 1.*

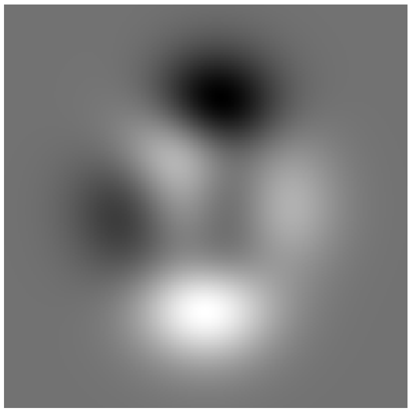
Figure 2: *Bump texture of the example 1.*

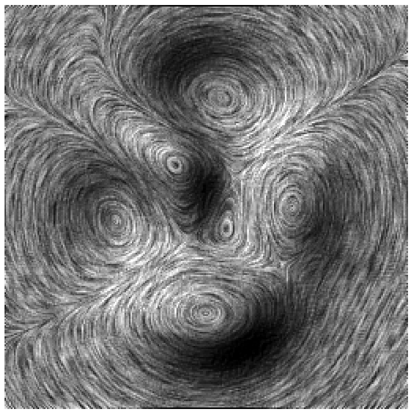
Figure 3: *Bumped (BLIC) texture of the example 1.*

3.1. Schema of the algorithm

The schema of the proposed algorithm is shown in Figure 4 and it can be spit in two steps.

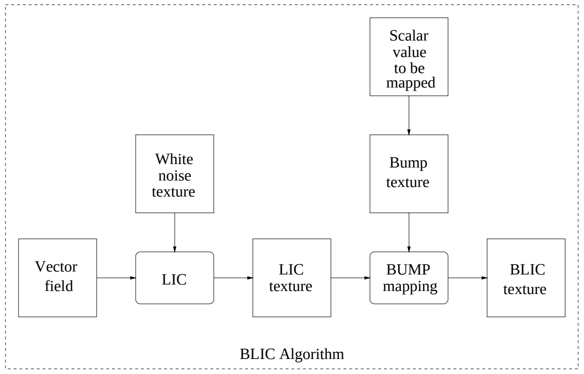

Figure 4: *BLIC schema.*

In the first step, a visualization of the vector field is computed by a texture-based algorithm such as LIC; the LIC algorithm receives in input a white noise texture and the vector field and gives in output the LIC texture.
The LIC texture is used in the second phase as input of the bump mapping process which uses a bump texture in order to produce the BLIC image. The bump map is a gray scale image of the same resolution of the LIC texture and it is computed according to the value of the scalar to be mapped in the texture.

4. Examples

In this Section the application of the proposed algorithm is shown on two examples. The first example shows the flow field corresponding to the natural development of a spatially evolving two-dimensional laminar mixing layer. A mixing layer originates in the merge of two parallel streams, each with a uniform velocity U_1 and U_2 ($U_1 > U_2$), both assumed in the same direction; the LIC texture for the example 1 is shown in Figure 5. In this case the bump texture (see Figure 6) is represented by the vorticity distribution along the vector field, and it allows to produce the BLIC texture of Figure 7 where the vortical structures, and above all, the vorticity present at the left part of the flow field are strongly enhanced with respect to the LIC texture.

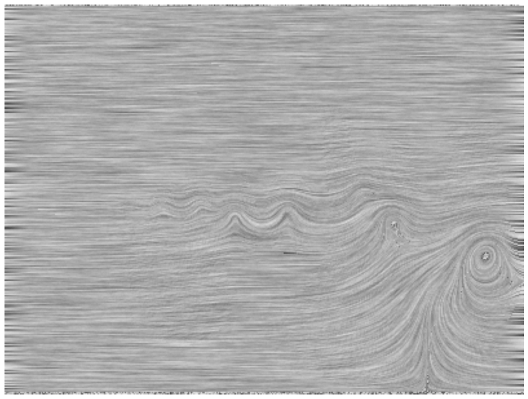

Figure 5: *LIC texture of the example 2.*

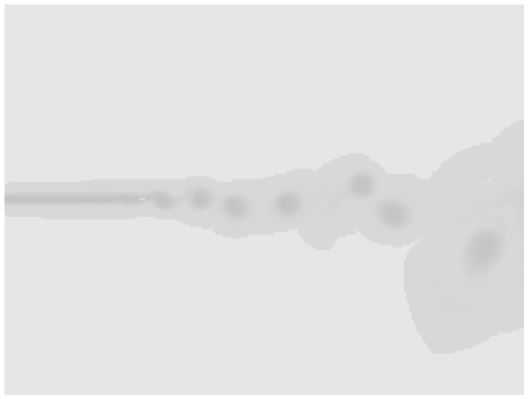

Figure 6: *Bump texture of the example 2.*

In the second example, the flow field past a backward facing step is visualized. The typical flow pattern displays the formation of a separated flow past the step edge, as well as the emergence of reattached flow downstream, on the lower wall (see Figure 9). In the same way of the first example, the bump texture represents the vorticity distribution along the vector field (see Figure 10) and it allows to obtain the bumped texture shown in Figure 11 that significantly enhances the details of the complex vortical structures.

5. Remarks

BLIC uses LIC as first step for the texture computation, but any other texture-based method could be employed; better

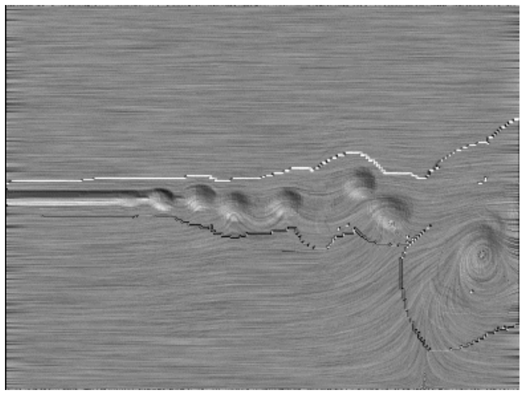

Figure 7: *Bumped (BLIC) texture of the example 2.*

results are obtained using techniques computing dense textures since the use of sparse textures like in OLIC can lead to a loss of detail (for instance, a small vorticity could not be displayed).
The examples presented in Section 4 have proved how the proposed method can enhance the understanding of a flow field allowing to better characterization of vortical structures. Moreover, the proposed method allows to map in the resulting texture the information concerning another scalar value beyond the information achievable from the classical texture synthesis techniques such as LIC; for instance, it is possible to map the trend information (as shown in Figure 3) by bumps or depressions or the vorticity information (see Figure 7 and Figure 11).
Different bump textures could be used to map additional scalar values; in this case, different output textures should be computed for each bump map. A significant improvement to the proposed algorithm could be the development of a methodology able to combine different bump textures in order to produce just one output texture where different scalar values can be mapped.
Colors are not required in the proposed algorithm. For this reason, it is always possible to map another scalar value; for example, the HSV model can be used, mapping the scalar value on the Hue component.
Finally, it is worth to stress the bump process can be carried out with an unnoticeable delay since almost all graphics workstations and several recent graphics cards for PCs perform the bump mapping in hardware.
On the other hand, a simple software implementation can be achieved using a ray tracer; a plane has to be defined and a material considering the LIC texture and the perturbed normal according to the bump texture must be assigned to the plane. For instance, POV-Ray[12] (the most popular freeware ray-tracer) allows to define a material as shown in Figure 8.

```
#declare Material1 =
   material
   {
      texture
      {
         pigment
         {
            image_map
            {
               gif   "lic.gif"
               once
            }
         }
         normal
         {
            bump_map
            {
               gif   "bump.gif"
               once
               interpolate 2
            }
            bump_size 20.0
         }
      }
   }
```

Figure 8: *Definition of a material in POV-Ray language.*

6. Conclusion

This paper presents a new texture-based technique to display vector fields. The bump mapping process is combined with classical texture-based synthesis algorithms in order both to provide a better characterization of interesting structures in the flow field and to map an additional scalar value in the output texture.
Bump mapping can be performed in hardware without delay by almost all graphics workstations and by the most part of the recent graphics cards for PCs. On the other hand, the examples presented have shown how the proposed method can significantly enhance the understanding of structures like vortices, allowing to map additional information coded by bump textures, with a real gain in multivariate visualization.

Acknowledgements

We thank Prof. Renzo Arina of the Department of Ingegneria Aeronautica e Spaziale - Politecnico di Torino for the examples 2 and 3 we have presented in Section 4.

References

1. J.L. Helman and L. Hesselink. Visualizing vector field topology in fluid flows. *IEEE Computer Graphics and Applications*, **11**:3, pp. 36–46, 1991. 1
2. G.M. Nielson, M. Magen and H. Müller. Scientific visualization overviews, methodologies, techniques. IEEE Computer Society Los Alamitos, California, 1997. 1
3. J.J. van Wijk. Spot noise-texture synthesis for data visualization. *ACM Computer Graphics (Proc. of SIGGRAPH '91)*, **25**, pp. 309–318, 1991. 1, 2
4. B. Cabral and L. Leedom. Imaging vector fields using line integral convolution. *ACM Computer Graphics (Proc. of SIGGRAPH'93)* **27**, pp. 263–270, 1993. 1, 2
5. D. Stalling and H.C. Hege. Fast and resolution independent line integral convolution. *ACM Computer Graphics (Proc. of SIGGRAPH'95)* pp. 249–256, 1995. 1, 2
6. R. Wegenkittl, E. Gröller, and W. Purgathofer. Animating flowfields: rendering of oriented line integral convolution. *Computer Animation'97*, pp. 15–21, 1997. 1, 2
7. R. Wegenkittl and E. Gröller. Fast oriented line integral convolution for vector field visualization via the internet. *Proceedings of IEEE Visualization '97*, pp. 309–316, 1997. 1, 2
8. H.W. Shen, C.R. Johnson, and K.L. Ma. Visualizing vector fields using line integral convolution and dye advection. *Proceedings of the ACM Symposium on Volume Visualization'96*, pp. 63–70, 102, 1996. 1
9. J. Blinn. Simulation of wrinkled surfaces. *ACM Computer Graphics (Proc. of SIGGRAPH '78)*, **12**, pp. 286–292, 1978. 1, 2
10. W.C. De Leeuw and J.J. van Wijk. Enhanced spot noise for vector field visualization. *Proceedings of IEEE Visualization'95*, pp. 233–239, 1995. 2
11. M. Zöckler, D. Stalling, and H.C. Hege. Parallel line integral convolution. *Parallel Computing*, **23**, pp. 975–989, 1997. 2
12. POV-Ray. http://www.povray.org. 4

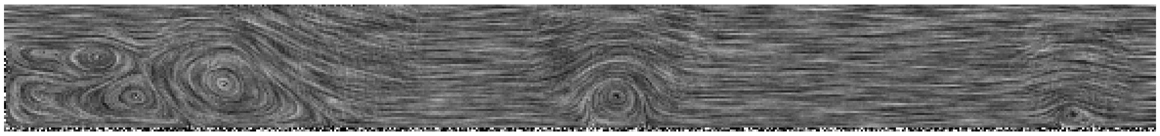

Figure 9: *LIC texture of the example 3.*

Figure 10: *Bump texture of the example 3.*

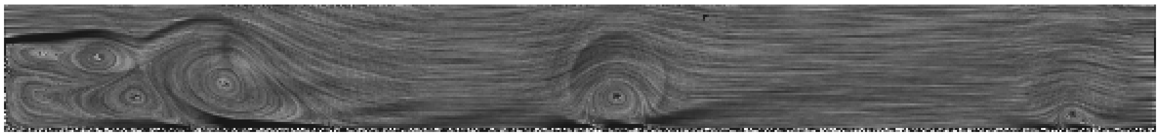

Figure 11: *Bumped (BLIC) texture of the example 3.*