TEMA Tend. Mat. Apl. Comput., **5**, No. 1 (2004), 17-25.

Visualização de Campos Vetoriais 3D Através da Iluminação de Linhas de Corrente

E.L. ARAÚJO[1], Pós Graduação, Laboratório Nacional de Computação Científica - LNCC, Rua Getúlio Vargas 333, Quitandinha, 25651-070 Petrópolis, RJ, Brasil

R. FARIAS[2], COPPE, Universidade Federal do Rio de Janeiro - UFRJ, Centro de Tecnologia, Bloco H, Sala 319, Ilha do Fundão, 21941-972 Rio de Janeiro, RJ, Brasil.

Resumo. Neste trabalho, apresentamos uma técnica para a visualização de linhas de corrente representando campos vetoriais 3D. A partir da dedução adequada de um vetor normal às linhas de corrente nos pontos onde pretendemos iluminar, utilizamos o suporte gráfico disponível na maioria das placas gráficas atuais. Através da aplicação de mapas de textura realizamos a iluminação das linhas de corrente aplicando o método de Phong[6] da computação gráfica.

Para uma melhor compreensão do método apresentamos o algoritmo que é utilizado para a criação das linhas de corrente a partir de um conjunto de dados discretos que representam o campo vetorial de uma simulação de um tornado.

1. Introdução

A representação visual de campos vetoriais é assunto de pesquisas em andamento em visualização científica. Um grande número de métodos sofisticados tem sido propostos para atacar este problema, variando desde o traçado de partículas [5],[8], até métodos baseados em aplicação de texturas [3], [2], [4]. Uma simples e poderosa técnica é descrever as linhas do campo, também chamadas de *curvas integrais* ou *linhas de corrente* Entretanto, usando este método o usuário é confrontado com alguns problemas. Um deles, é que o desenho de retas não oferece uma boa percepção espacial da imagem, o que pode ser resolvido limitando o número de linhas exibidas. Porém, o número limitado de linhas de corrente para conjuntos de dados grandes, fornece uma percepção pobre do campo vetorial. Neste trabalho apresentamos um algoritmo que pode superar este problema.

É bem conhecido em computação gráfica, que o realismo das imagens geradas por computador dependem de um modelo que aproxime o mais realisticamente possível a interação da luz com os objetos em uma cena. Os efeitos de iluminação fornecem os mais importantes detalhes da percepção espacial. Conseqüentemente, muitos

[1]edson@lncc.br.
[2]rfarias@cos.ufrj.br

pesquisadores têm trabalhado no intuito de desenvolver modelos de iluminação e reflexão em computação gráfica. Um modelo amplamente usado, com relativa complexidade computacional e alto realismo é o modelo de reflexão de Phong[6], que assume as fontes de luz como sendo pontuais e aproxima os mais importantes termos de reflexão por expressões simples. Tradicionalmente este modelo é aplicado para superfícies. Entretanto, a maioria dos computadores atuais, não oferece suporte em hardware para a aplicação deste modelo em retas. Desta forma a maioria dos cálculos da iluminação de linhas são feitos via software. Apresentaremos neste trabalho, uma forma rápida e correta de iluminação de linhas de corrente, explorando a aplicação de textura disponível nos modernos computadores, em nível de hardware. Na próxima seção discutimos como é feito o traçado de linhas de corrente a partir de um conjunto de dados discretos. Em seguida, na seção 3, exibimos a nossa dedução para o vetor normal que usaremos para efetuar a iluminação das linhas de corrente. Na seção 4, discutimos a utilização do hardware gráfico para os cálculos necessários na iluminação. Nas seções 5 e 6, exibimos alguns resultados e nossas conclusões a respeito do algoritmo apresentado.

2. Traçado de Linhas de Corrente

O cálculo do caminho descrito por uma partícula sob a influência de um campo vetorial é baseado na integração numérica da equação diferencial

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}(\mathbf{x}), \tag{2.1}$$

onde t denota tempo, $\mathbf{x}$ a posição da partícula e $\mathbf{v}(\mathbf{x})$ a velocidade do campo. A posição inicial $\mathbf{x}_0$ da partícula fornece a condição inicial:

$$\mathbf{x}(t_0) = \mathbf{x}_0.$$

A solução é uma seqüência de posições $(\mathbf{x}(t_0), \mathbf{x}(t_1), ...)$ da partícula, que representa o caminho da partícula ou, em outras palavras, a linha de corrente que passa por $\mathbf{x}_0$.

Vários esquemas podem ser usados para integrar a equação (2.1). Um esquema comumente usado é o *método de integração de segunda ordem de Runge-Kutta com passo adaptativo.* Este método tenta a cada integração sucessiva, modificar o comprimento do passo de integração de modo que o erro de aproximação se mantenha constante. Considere $\mathbf{x}_n$ sendo a posição atual da partícula em $t = t_n$ e $\mathbf{x}_{n+1}$ a próxima posição em t_{n+1}. O seguinte pseudo-código determina x_{n+1} e calcula o passo h a ser usado na próxima integração, que manterá o erro ε, pré-definido, constante:

v_n = v(x_n);
x_{mid}= x_n + 0.5*h*v_n;
x_{full}= x_n + h*v_n;
v_{mid}= v(x_{mid});
x_{n+1}= x_{mid} + 0.5*h*v_{mid};
error=$|x_{n+1}-x_n|$
if(error$> \varepsilon$)
    h_{new}= 0.9*h_{old}*$|\frac{\varepsilon}{error}|^{\frac{1}{2}}$
else
    h_{new}= 0.9*h_{old}*$|\frac{\varepsilon}{error}|^{\frac{1}{3}}$.

O passo inicial h pode ser escolhido como $h = \dfrac{c}{\max(v_n)}$, onde $0 \leq c \leq 1$ e $\max(v_n)$ representa a máxima coordenada de v_n. A constante c controla o tamanho do passo da partícula. Maiores detalhes a respeito deste esquema de integração podem ser encontrados em [7].

Um algoritmo para o traçado de linhas de corrente deve realizar os seguintes passos. Primeiro, uma busca sobre as células do conjunto para determinar qual contém a posição inicial da partícula. Determinar a velocidade neste ponto, isto é feito através da interpolação das velocidades nos vértices da célula. Então, um passo de integração calcula a nova posição da partícula. Novamente, uma busca é realizada para encontrar a célula que contém a nova posição. O processo de busca, interpolação e integração é repetido até que a partícula deixe o gride ou atinga um certo número máximo de passos de integração. Este algoritmo pode ser escrito em pseudo-código da seguinte forma:

```
busca da célula contendo a posição inicial
while(partícula está no grid && no.  de integrações < N){
    determinar a velocidade na posição corrente
    calcular a nova posição
    buscar célula contendo a nova posição
}
```

3. Iluminação de Linhas em $\mathbb{R}^3$

Superfícies podem ser caracterizadas localmente por um vetor normal **N** exterior. Este vetor normal detém uma importância significativa quando descrevendo a interação da luz com elementos da superfície. O bem conhecido modelo de reflexão de Phong, determina a intensidade de cor em cada um dos pontos de uma superfície através da seguinte equação:

$$\begin{aligned} I &= I_{amb} + I_{diff} + I_{spec} \\ &= k_a + k_d\,(\mathbf{L}\cdot\mathbf{N}) + k_s\,(\mathbf{V}\cdot\mathbf{R})^n\,, \end{aligned} \tag{3.1}$$

onde L, V, R são vetores unitários que denotam a direção da luz, direção de vista, reflexão da luz em torno do vetor N unitário normal externo à superfície, respectivamente.

O primeiro termo, representa a intensidade de luz devido a múltiplas reflexões no ambiente. O segundo termo descreve a reflexão difusa segundo a *Lei de Lambert*. A intensidade de luz difusa não depende da direção do observador, ou seja, os efeitos da reflexão difusa são iguais em todas as direções. O último termo em (3.1) descreve a reflexão especular sobre a superfície. Reflexões especulares, são centradas em torno do vetor de reflexão $\mathbf{R}$, a largura do reflexo especular é controlada pelo expoente n.

A aplicação deste modelo para a iluminação de linhas torna-se difícil dada a infinita quantidade de vetores normais à uma linha. Entretanto, podemos tomar como vetor $\mathbf{N}$ normal à linha, aquele que é coplanar aos vetores de luz $\mathbf{L}$ e o vetor $\mathbf{T}$ tangente à linha (veja Figura 1).

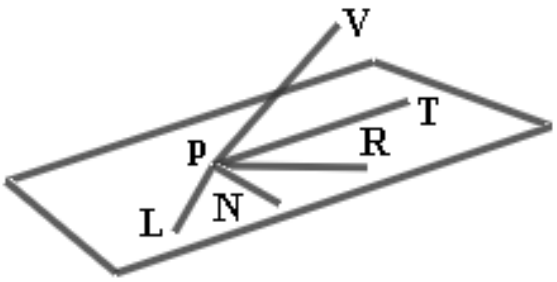

Figura 1: Escolha do vetor normal.

Desta forma, podemos calcular a componente de reflexão difusa para linhas do mesmo modo que se faz para superfícies, usando (3.1). Da mesma forma, entre todos os vetores de reflexão possíveis escolhemos aquele que é coplanar a $\mathbf{L}$ e a $\mathbf{T}$. Novamente, usando (3.1) podemos calcular a componente de reflexão especular para linhas em $\mathbb{R}^3$.

Em vez de calcular os vetores escolhidos acima, explicitamente, gostaríamos de expressar as componentes de reflexão difusa e especular presentes em (3.1) em função de $\mathbf{L}, \mathbf{T}$ e $\mathbf{V}$ somente, ou seja, sem termos que calcular $\mathbf{N}$. Assim sendo, como $\mathbf{L}, \mathbf{T}$ e $\mathbf{N}$ são coplanares, podemos escrever $\mathbf{L}$ como combinação linear de $\mathbf{T}$ e $\mathbf{N}$ da seguinte forma:

$$\mathbf{L} = (\mathbf{L} \cdot \mathbf{T})\mathbf{T} + (\mathbf{L} \cdot \mathbf{N})\mathbf{N}$$

e, conseqüentemente, temos:

$$\mathbf{L} \cdot \mathbf{N} = \sqrt{1 - (\mathbf{L} \cdot \mathbf{T})^2} \ . \tag{3.2}$$

Usando argumentos semelhantes, podemos também expressar $\mathbf{V} \cdot \mathbf{R}$, responsável pela componente especular em (3.1), em termos de $\mathbf{L}$ e $\mathbf{T}$. Primeiro, observe que, sendo $\mathbf{R}$ a reflexão de $\mathbf{L}$ em torno de $\mathbf{N}$, podemos afirmar que

$$\mathbf{R} = (\mathbf{L} \cdot \mathbf{N})\mathbf{N} - (\mathbf{L} \cdot \mathbf{T})\mathbf{T},$$

de modo que

$$\begin{aligned}\mathbf{V}\cdot\mathbf{R} &= \mathbf{V}\cdot[(\mathbf{L}\cdot\mathbf{N})\mathbf{N}-(\mathbf{L}\cdot\mathbf{T})\mathbf{T}] \\ &= (\mathbf{L}\cdot\mathbf{N})\,(\mathbf{V}\cdot\mathbf{N})-(\mathbf{L}\cdot\mathbf{T})\,(\mathbf{V}\cdot\mathbf{T}) \\ &= \sqrt{1-(\mathbf{L}\cdot\mathbf{T})^2}\,(\mathbf{V}\cdot\mathbf{N})-(\mathbf{L}\cdot\mathbf{T})\,(\mathbf{V}\cdot\mathbf{T})\,.\end{aligned} \tag{3.3}$$

Aqui, trocamos $\mathbf{L}\cdot\mathbf{N}$ por (3.2). Ainda precisamos expressar o termo $\mathbf{V}\cdot\mathbf{N}$ em (3.3) também, em função de $\mathbf{L},\mathbf{T}$ e $\mathbf{V}$, apenas. Para isso, observe que o vetor $\mathbf{V}$, pode ser escrito da seguinte forma:

$$\mathbf{V} = (\mathbf{V}\cdot\mathbf{N})\,\mathbf{N} + (\mathbf{V}\cdot\mathbf{T})\,\mathbf{T} + \left[\mathbf{V}\cdot\tfrac{(\mathbf{L}\times\mathbf{T})}{\|\mathbf{L}\times\mathbf{T}\|^2}\right](\mathbf{L}\times\mathbf{T})\ .$$

Donde segue-se que:

$$\mathbf{V}\cdot\mathbf{N} = \sqrt{1-(\mathbf{V}\cdot\mathbf{T})^2-\left[\tfrac{\mathbf{V}\cdot(\mathbf{L}\times\mathbf{T})}{\|\mathbf{L}\times\mathbf{T}\|}\right]^2}\ . \tag{3.4}$$

Das propriedades do produto vetorial entre vetores em $\mathbb{R}^3$, temos que

$$\|\mathbf{L}\times\mathbf{T}\| = \|\mathbf{L}\|\,\|\mathbf{T}\|\,\mathrm{sen}\phi,$$

onde ϕ é o ângulo entre os vetores $\mathbf{L}$ e $\mathbf{T}$. Usando os fatos de que $\mathbf{L}$ e $\mathbf{T}$ são unitários e $\mathbf{L}\cdot\mathbf{T}=\cos\phi$, segue-se que

$$\|\mathbf{L}\times\mathbf{T}\| = \mathrm{sen}\phi = \sqrt{1-\cos^2\phi} = \sqrt{1-(\mathbf{L}\cdot\mathbf{T})^2}\ .$$

Voltando à eq. (3.4), teremos

$$\begin{aligned}\mathbf{V}\cdot\mathbf{N} &= \sqrt{1-(\mathbf{V}\cdot\mathbf{T})^2-\frac{[\mathbf{V}\cdot(\mathbf{L}\times\mathbf{T})]^2}{1-(\mathbf{L}\cdot\mathbf{T})^2}} \\ &= \frac{\sqrt{[1-(\mathbf{L}\cdot\mathbf{T})^2]\,[1-(\mathbf{V}\cdot\mathbf{T})^2]-[\mathbf{V}\cdot(\mathbf{L}\times\mathbf{T})]^2}}{\sqrt{1-(\mathbf{L}\cdot\mathbf{T})^2}}\ .\end{aligned} \tag{3.5}$$

E, substituindo (3.5) em (3.3) teremos finalmente

$$\mathbf{V}\cdot\mathbf{R} = \sqrt{[1-(\mathbf{L}\cdot\mathbf{T})^2]\,[1-(\mathbf{V}\cdot\mathbf{T})^2]-[(\mathbf{V}\times\mathbf{L})\cdot\mathbf{T}]^2}-(\mathbf{L}\cdot\mathbf{T})(\mathbf{V}\cdot\mathbf{T})\ .$$

4. Renderizando Linhas com Iluminação

Embora as equações de iluminação para linhas e superfícies pareçam as mesmas, o uso do hardware padrão para iluminação de linhas é prejudicado, por que, para cada nova direção de vista, um vetor normal adequado tem que ser calculado sem utilizar o hardware gráfico. Nesta seção exibimos uma forma de calcular as equações (3.2) e (3.4) usando a aplicação de textura disponível na maioria dos hardwares gráficos modernos. Desta forma, evitamos o cálculo explícito do vetor normal a cada mudança de vista.

4.1. Aplicação de Textura

Assumimos nesta seção, ter disponível uma API gráfica semelhante ao OpenGL. Nesta API, a aplicação de textura é feita através de um mapa de cores $\mathbf{P}$ e uma matriz $\mathbf{M}$ de textura 4×4. O mapa de textura $\mathbf{P}$, é na verdade uma outra matriz n-dimensional, onde $1 \leq n \leq 3$, que contém uma cor em cada uma das suas entradas. Para determinar a cor que um ponto $\mathbf{p}$ receberá da textura, especifica-se um vetor em coordenadas homogêneas $\mathbf{v}$, que multiplicado pela matriz de textura $\mathbf{M}$, tem como resposta um outro vetor $\mathbf{Mv}$ cujas n primeiras coordenadas são utilizadas como índices de busca no mapa de textura $\mathbf{P}$. Esta transformação de textura é o ponto chave para tornar possível a utilização do hardware gráfico na iluminação de linhas em $\mathbb{R}^3$.

4.2. Reflexão Difusa

Observando a equação (3.2), podemos ver que a componente de reflexão difusa de um segmento de linha é uma função de $(\mathbf{L} \cdot \mathbf{T})$. Então, especificando como vetor de textura t_0 o vetor tangente $\mathbf{T}$ em cada vértice do segmento, este produto interno pode ser calculado em hardware através da seguinte matriz de textura:

$$\mathbf{M} = \frac{1}{2} \begin{pmatrix} L_1 & L_2 & L_2 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} .$$

A primeira componente do vetor homogêneo transformado $\mathbf{t} = \mathbf{Mt}_0$ será

$$t_1 = \frac{1}{2}(\mathbf{L} \cdot \mathbf{T} + 1) .$$

Observe que $0 \leq t_1 \leq 1$. Logo, este valor pode ser usado como índice dentro de uma matriz de textura unidimensional $\mathbf{P}(t_1)$. O valor do mapa de textura $\mathbf{P}$ no local indicado por t_1 é escolhido de tal forma que se obtenha a componente de luz difusa, ou seja,

$$\mathbf{P}(t_1) = k_d \sqrt{1 - (2t_1 - 1)^2} .$$

Desta forma, através do uso da matriz e do mapa de textura, teremos calculado em hardware a correta iluminação difusa do segmento de reta.

4.3. Reflexão Especular

A componente de reflexão especular não depende apenas de $\mathbf{L} \cdot \mathbf{T}$, mas também de $\mathbf{V} \cdot \mathbf{T}$ e $(\mathbf{V} \times \mathbf{L}) \cdot \mathbf{T}$, como pode ser visto em (4.1). Para calcular estes dois produtos internos adicionais, inicializamos a segunda e terceira linhas da matriz de textura com os vetores $\mathbf{V}$ e $\mathbf{W} = (\mathbf{V} \times \mathbf{L})$:

$$\mathbf{M} = \frac{1}{2} \begin{pmatrix} L_1 & L_2 & L_2 & 1 \\ V_1 & V_2 & V_3 & 1 \\ W_1 & W_2 & W_3 & 1 \\ 0 & 0 & 0 & 2 \end{pmatrix} .$$

Assim, procedendo como na seção anterior, as componentes do vetor homogêneo transformado $\mathbf{t} = \mathbf{M}\mathbf{t}_0$ serão

$$\begin{aligned} t_1 &= \tfrac{1}{2}\left(\mathbf{L}\cdot\mathbf{T}+1\right), \\ t_2 &= \tfrac{1}{2}\left(\mathbf{V}\cdot\mathbf{T}+1\right), \\ t_3 &= \tfrac{1}{2}\left((\mathbf{V}\times\mathbf{L})\cdot\mathbf{T}+1\right). \end{aligned} \tag{4.1}$$

E como anteriormente, $0 \leq t_1, t_2, t_3 \leq 1$, o que torna possível a utilização de t_1, t_2, t_3 como índices para um mapa de textura tridimensional $\mathbf{P}(t_1, t_2, t_3)$, onde

$$\begin{aligned} \mathbf{P}(t_1,t_2,t_3) \;=\; & k_a + k_d\sqrt{1-(2t_1-1)^2} \\ & +k_s\{\sqrt{[1-(2t_1-1)^2]\,[1-(2t_2-1)^2]-(2t_3-1)^2} \\ & -(2t_1-1)\,(2t_2-1)\}^n. \end{aligned}$$

Observe que a função $\mathbf{P}(t_1, t_2, t_3)$ já inclue as componentes de reflexão ambiente, difusa e especular, de forma que a aplicação da textura 3D produzida para esta função ocasiona a correta iluminação de um segmento de reta utilizando o modelo de Phong, via hardware.

4.4. Controlando a Intensidade do Brilho

Foi apontado por [1] que existe, em geral, um problema quando iluminando objetos com codimensão maior que um. A intensidade de iluminação global de uma imagem aumenta e torna-se mais uniforme, assim deturpando a percepção espacial. No nosso caso, linhas em $\mathbb{R}^3$, é sugerido em [1] a utilização de um expoente para a componente difusa, para compensar este problema, assim

$$I_{diff} = k_g\left(\mathbf{L}\cdot\mathbf{N}\right)^p,$$

onde $p = 4.8$ é proposto.

5. Implementação e Resultados

O algoritmo apresentado neste trabalho foi implementado em C++. O rendering é feito com o auxílio da biblioteca gráfica OpenGL. O código é composto de aproximadamente 1200 linhas, disponível dos autores.

Aplicamos o método para visualizar campos vetoriais definidos em grides regulares e abaixo exibimos algumas figuras obtidas.

6. Conclusões

Apresentamos, neste trabalho, uma técnica para a utilização do hardware gráfico padrão para efetuar a iluminação das linhas de corrente presentes em campos vetoriais 3D utilizando o modelo de Phong[6] de computação gráfica. A aceleração

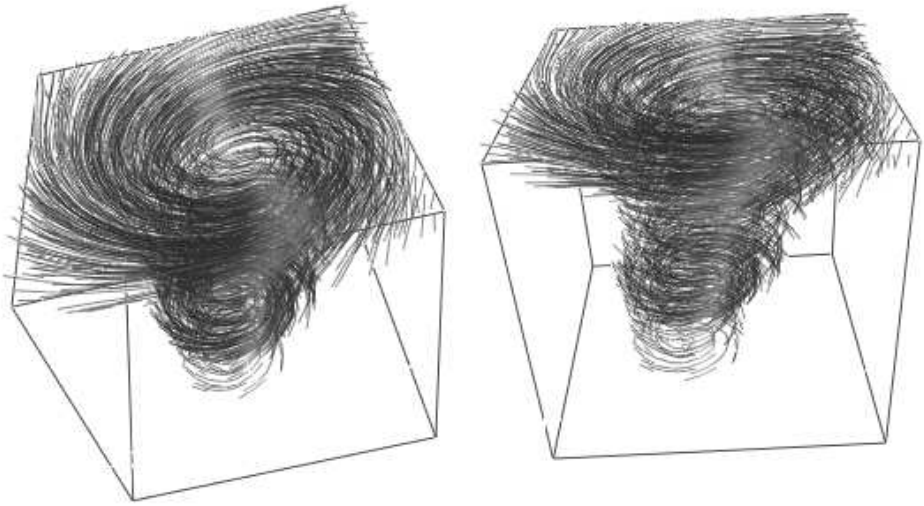

Figura 2: Simulação numérica de um tornado. É possível ter, nestas imagens, uma percepção espacial das linhas desenhadas. Dados cedidos por R. Carwfis.

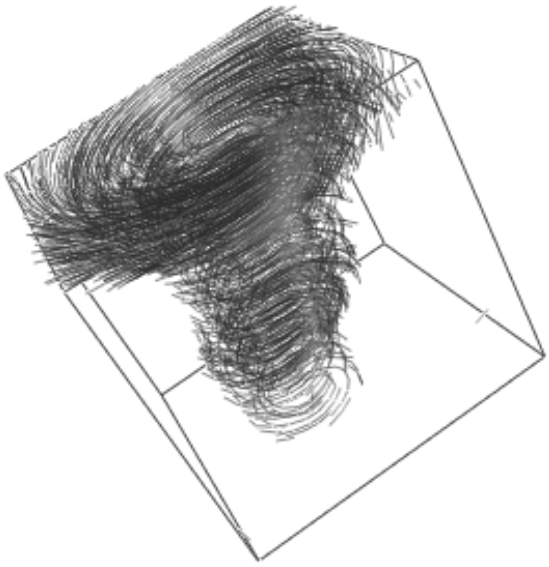

Figura 3: Tornado visto de outro ângulo.

gráfica é utilizada para agilizar a iluminação. Fornecemos assim uma maneira de efetuar a iluminação de forma rápida e eficiente.

Abstract. We present a new technique for visualization of three dimensional vector field streamlines. From a suitable deduction of a normal vector to the streamlines at each point to be illuminated, we use the graphics hardware support, available to most of the modern graphics cards nowadays, to perform the illumination of the streamlines using Phong model, by using texture maps.

To better understand the method, we show the algorithm used to create the streamlines out of a discrete data representing the vector field of a tornado simulation.

Referências

[1] D.C. Banks, Illumination in Diverse Codimensions, em "Proc. ACM Siggraph '94", Orlando, Florida, *Computer Graphics Ann. Conf. Series*, pp. 327-334, July, 1994.

[2] B. Cabral e L. Leedom, Imaging Vector Fields Using Line Integral Convolutio, em "Proc. ACM Siggraph '93", Anahein, California, *Computer Graphics*, vol. 27, pp. 263-272, Aug. 1993.

[3] R. Crawfis e N. Max, Textured Splats For 3D Scalar and Vector Field Visualization, em "Proc. Visualization '93" (G. Nielson and D. Bergeron, eds.), pp. 261-272. IEEE CS Press, 1993.

[4] L.K. Forsell, Visualization Flow Over Curvilinear Grid Surfaces Using Line Integral Convolution, em "Proc. of IEEE Visualization '94", pp. 240-247, 1994.

[5] A.J.S. Hin e F.H. Post, Visualization of Turbulent Flow with Particles, "Proc. Visualization '93", pp. 46-52, IEEE CS Press, 1993.

[6] B.-T. Phong, Illumination for Computer Generated Pictures, *Comm. ACM*, pp. 311-317, June, 1975.

[7] Press, William, et al., Numerical Recipes in C (pp. 574-578), Cambrige University Press, Cambrige, 1988.

[8] J.J. van Wijk, Rendering Surface-Particles, em "Proc. Visualizations '92", pp. 54-61, IEEE CS Press, 1992.

[9] M. Zöckler, D. Stalling e H.C. Hege, Interactive Visualization of 3D-vector Fields using Illuminated streamlines, em "Proceedings of IEEE Visualizations '96´´, pp. 107-113, 1996.