Projeto Final de Programação

Gerador Automático de Funções de Transferência

Orientador: Waldemar Celes

Aluno: Rustam Mesquita

Matrícula: 1512345

Julho / 2016

1. Introdução

Em visualização volumétrica, função de transferência é o nome dado ao mapeamento feito entre dados do volume e atributos ópticos, como cor e opacidade. Esse mapeamento permite destacar regiões de interesse, gerando uma visualização mais clara do volume. No entanto, a obtenção manual de uma boa função de transferência exige o conhecimento de como os dados variam no volume e paciência, já que esse método é consistido basicamente de tentativa e erro.

Tendo em vista a necessidade de gerar funções de transferência para a dissertação do mestrado e a dificuldade natural de realizar esta tarefa manualmente, optou-se por implementar um método de geração automática de funções de transferência como projeto final de programação. Como a renderização de volumes é por si só uma outra área de estudo, o projeto será integrado a um visualizador volumétrico a parte, de código aberto. Desta forma, não só o projeto está altamente alinhado à dissertação, como também será útil para os usuários do visualizador ao qual será integrado.

O projeto fará uso do método de geração automática descrito no artigo “Semi-automatic Generation of Transfer Functions for Direct Volume Rendering”, Kindlmann e Durkin, 1998. Portanto, os resultados serão validados pelo uso de volumes utilizados no artigo citado.

1. Requisitos Não Funcionais

* O programa deve ser desenvolvido na linguagem C / C++.
* O programa deve ser integrado a um visualizador volumétrico.

1. Requisitos Funcionais

* O programa deve gerar como saída uma função de transferência, tendo como entrada um volume de dados.
* O programa deve passar ao visualizador a função de transferência gerada.
* O programa não deve interferir no uso do visualizador por parte do usuário, de forma que este pode fazer uso de uma função de transferência própria.
* As funções de transferências geradas devem ser arquivos de extensão “tf1d” e seguir o seguinte formato:
  + <comentário>
  + <algarismo zero>
  + <quantidade de atributos de cor>
  + <primeiro atributo de cor>

⋮

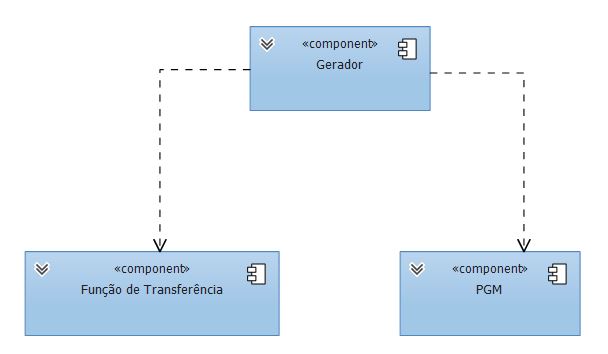
* + <último atributo de cor>
  + <quantidade de atributos de opacidade>
  + <primeiro atributo de opacidade >

⋮

* + <último atributo de opacidade >
* Cada atributo de cor deve ser especificado por 4 números espaçados variando de 0 a 255, onde os 3 primeiros representam a cor em rgb e o último representa o valor de intensidade a qual a cor está associada.
* Cada atributo de opacidade deve ser especificado por 2 números espaçados: o valor da opacidade e o valor de intensidade a qual a opacidade está associada. A opacidade deve variar de 0 a 1 e a intensidade de 0 a 255.

1. Arquitetura

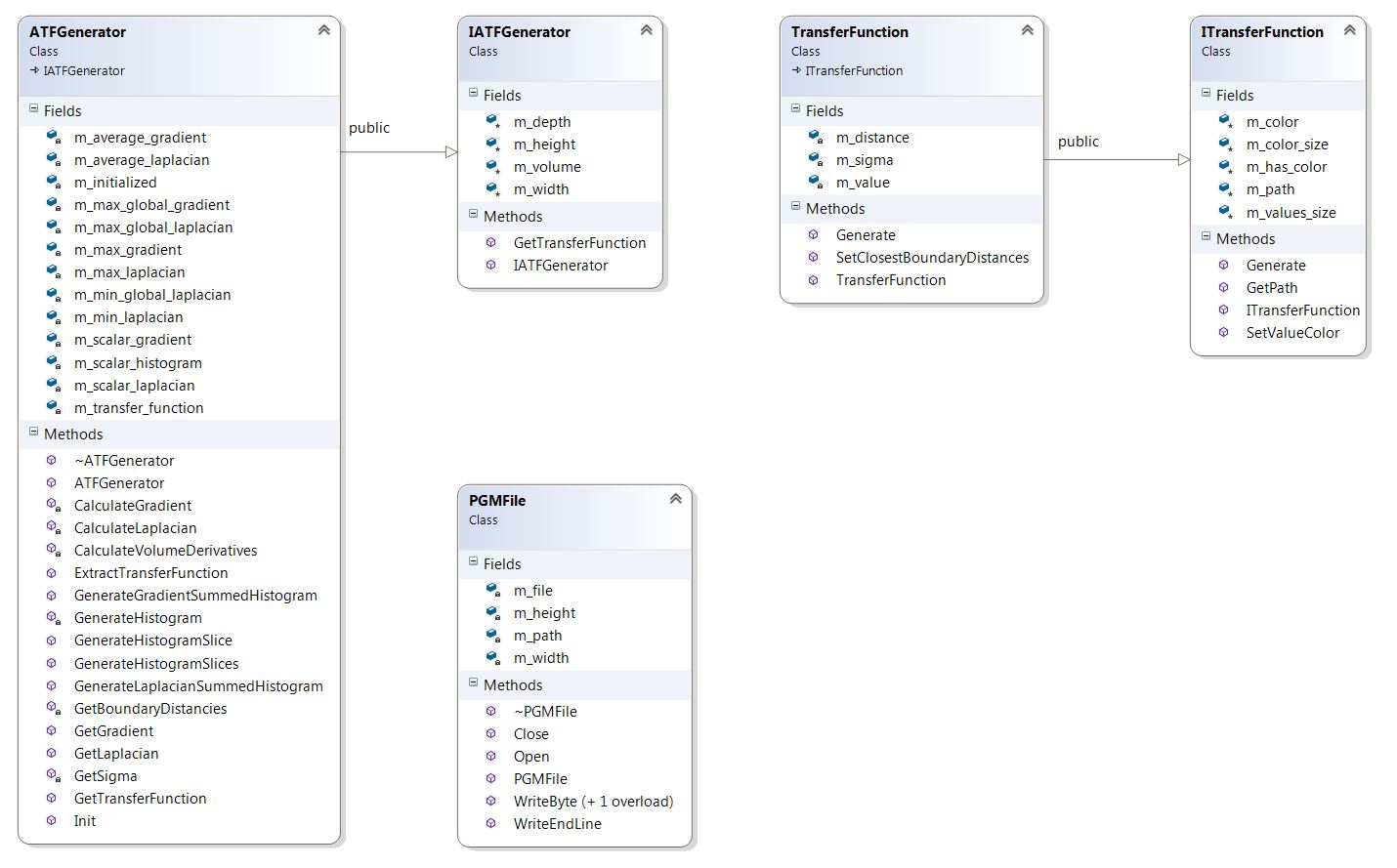
O projeto foi desenvolvido em 3 módulos: Gerador, Função de Transferência e PGM. O módulo PGM fornece uma interface para gerar imagens no formato “PGM”. O módulo Função de Transferência fornece uma interface para gerar funções de transferência. E o módulo Gerador é responsável por analisar o volume de entrada e a partir dele extrair uma função de transferência. A **Figura 1** ilustra a relação de dependência entre os módulos.



**Figura 1:** Diagrama de componentes do projeto.

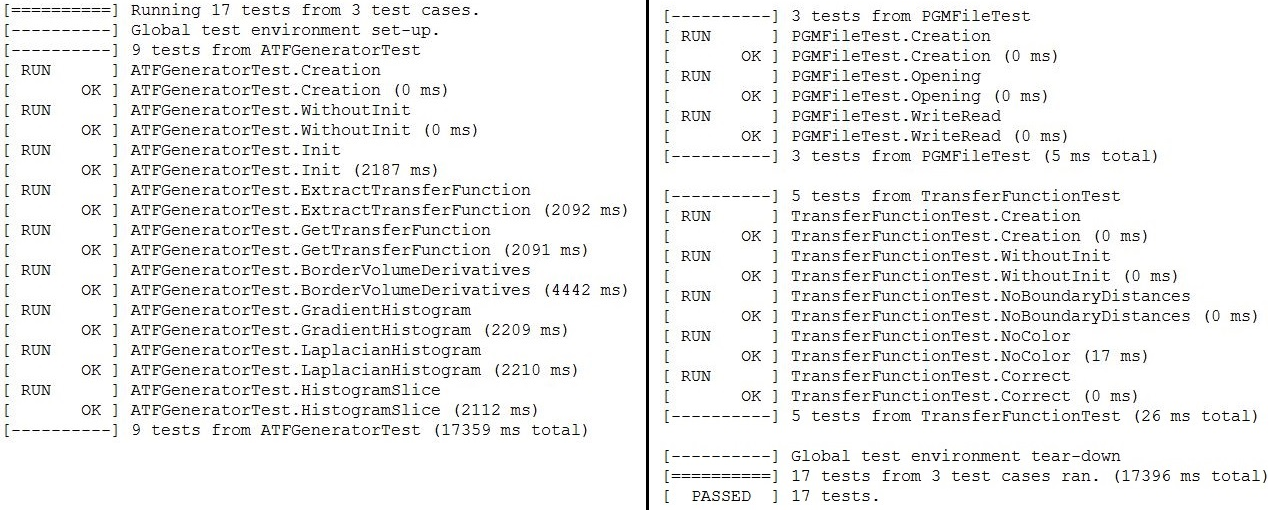
O programa foi projetado de forma a permitir a implementação de outros métodos que geram funções de transferência automaticamente. Para atingir esse objetivo, os módulos Gerador e Função de Transferência implementam interfaces que especificam a troca de informações entre o visualizador e o programa. Desta forma, novas técnicas podem ser utilizadas a partir de novas implementações das mesmas interfaces.

O módulo Gerador é implementado nas classes IATFGenerator e ATFGenerator, o módulo Função de Transferência é implementado nas classes ITransferFunction e TransferFunction, e o módulo PGM é implementado na classe PGMFile. A **Figura 2** mostra o relacionamento e a composição das classes.



**Figura 2:** Diagrama de classes do projeto.

1. Testes



**Figura 3:** Resultado dos testes unitários.

Foram realizados testes unitários com o auxílio da biblioteca de GTest. O relatório de execução de todos os testes está indicado na **Figura 3**, que mostra os resultados parciais e finais para cada caso de teste, bem como o resultado geral dos testes.

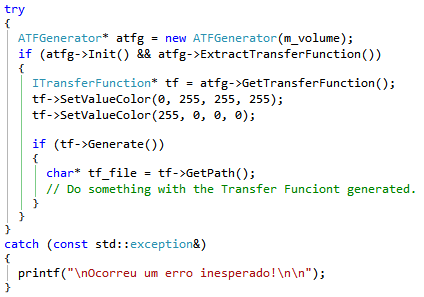
1. Manual do Usuário

A extração automática de uma função de transferência precisa seguir a seguinte sequência de passos: inicialização, extração, obtenção e geração.

A partir de uma instancia da classe **ATFGenerator**, a inicialização é feita através da chamada ao método **Init**. Esse método aloca toda a memória necessária para a obtenção da função de transferência, analisa o volume e gera seu histograma. Após essa chamada, o usuário já pode visualizar as fatias do histograma através do método **GenerateHistogramSlice** ou visualizar o histograma acumulado através dos métodos **GenerateGradientSummedHistogram** e **GenerateLaplacianSummedHistogram**. É importante ressaltar que qualquer chamada só pode ser feita se o método **Init** retornar verdadeiro.

A extração da função de transferência é feita através do método **ExtractTransferFunction**. Baseado no histograma gerado internamente esse método cria uma função de transferência. Esta deve ser obtida através do método **GetTransferFunction**, que retorna uma instancia para a classe I**TransferFunction**. A partir da função de transferência obtida o usuário precisa configurar ao menos duas cores a serem interpoladas. Essa configuração é feita através do método **SetValueColor**.

Uma vez extraída a função de transferência e tendo escolhido suas cores é possível gerá-la. Isso é feito através do método **Generate.** Esse método gera um arquivo de função de transferência de extensão “tf1d”, cujo caminho completo pode ser obtido pelo método **GetPath.** A **Figura 4** mostra um exemplo de como utilizar a API. Mais detalhes sobre as classes e os métodos estão na documentação, disponível na pasta “doc”.



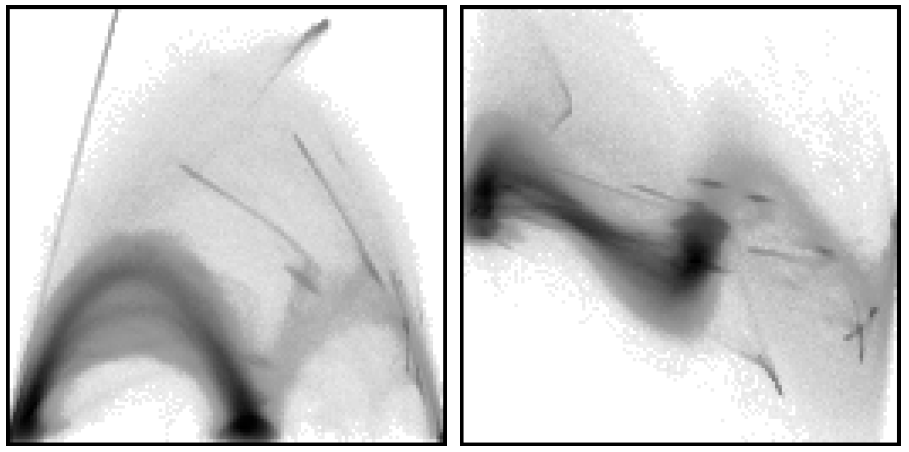
**Figura 4:** Exemplo de uso da biblioteca.

1. Resultados

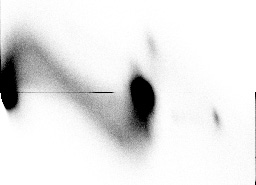
As figuras 5 e 6 mostram, respectivamente, os histogramas acumulados pelo gradiente e pelo laplaciano do volume de uma peça mecânica. A **Figura 5** é resultado do trabalho “Semi-automatic Generation of Transfer Functions for Direct Volume Rendering”, Kindlmann e Durkin, 1998, enquanto que a **Figura 6** é resultado do trabalho desenvolvido neste projeto.

Os gradientes acumulados revelam arcos que permitem a identificação de interfaces entre valores diferentes do volume. Isto é, cada arco representa uma fronteira, ou um contorno. Isso faz com que essas imagens sejam boas métricas para validar os resultados obtidos.

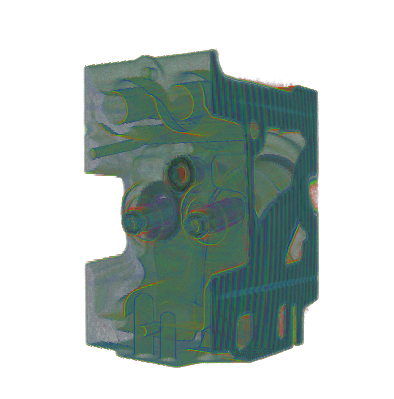
Tanto na figura 6.a quanto na figura 6.b é possível identificar os principais arcos correspondentes aos das figuras 5.a e 5.b. Por esse resultado é possível pressupor que a principal fronteira do volume foi identificada e que, portanto, a função de transferência gerada deve permitir ao menos uma visualização que permita a identificação do modelo. O pressuposto pode ser verificado através da **Figura 7.**



**Figura 5:** a) histograma acumulado pelo gradiente. b) histograma acumulado pelo laplaciano. Ambas são resultado do trabalho de Kindlmamn e Durkin, sob o volume de uma peça mecânica.



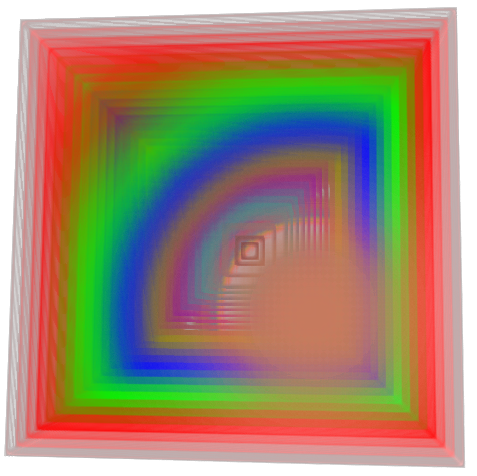
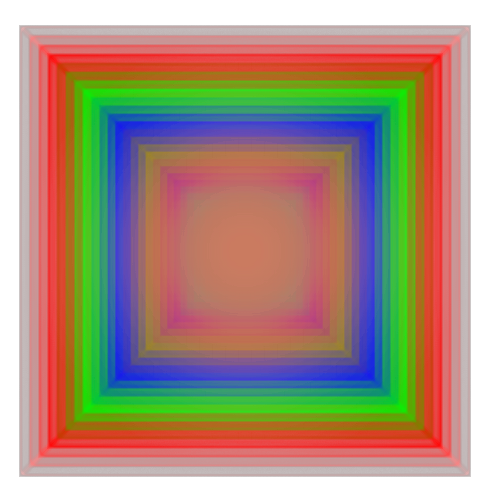
**Figura 6:** a) histograma acumulado pelo gradiente. b) histograma acumulado pelo laplaciano. Ambas são resultado deste projeto, sob o mesmo volume de uma peça mecânica, utilizado por Kindlmamn e Durkin.



**Figura 7:** Visualização do volume de peça mecânica, utilizando a função de transferência gerada automaticamente pela avaliação do volume.

No entanto, as figuras 5 e 6 não são idênticas. É possível perceber que alguns arcos e contornos não foram reproduzidos por este trabalho. Isso se deve à diferença entre a gama de valores utilizados para representar os gradientes e laplacianos do volume. Esses valores podem ser muito grandes e dispersos, devido principalmente à presença de ruído nos dados. Por isso, no trabalho utilizado como referência, Kindlmamn e Durkin deixam claro que não devem ser utilizados todos os valores de primeira e segunda derivada na geração do histograma. Isso previne que valores significativos, responsáveis por revelar as fronteiras, sejam comprimidos no histograma, ganhando um peso menor na avaliação do volume.

Apesar da justificativa acima, Kindlmamn e Durkin não explicitam qual método é utilizado para limitar os valores de gradiente e laplaciano. O artigo relata que esta definição ainda exige estudo e que por ora foi utilizado um “palpite educado”. Neste projeto optou-se por não efetuar o corte nos valores e estudar os resultados para posteriormente aplicar um corte apropriado. Esta decisão é refletida na diferença entre os resultados das imagens 5 e 6.



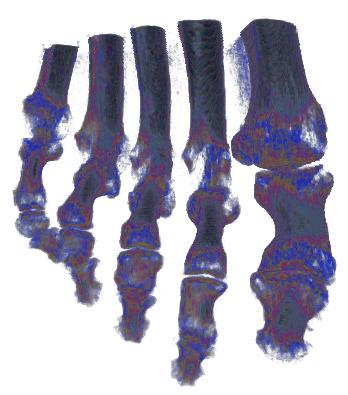
**Figura 8:** Volume analítico diversos cubos.

Apesar da escolha feita é possível verificar um resultado satisfatório na geração automática de funções transferências (sabendo que a qualidade desse resultado pode variar com a dispersão dos dados do volume analisado). Um modo de verificar esse resultado satisfatório é avaliar a saída do programa para uma entrada conhecida. Com essa intenção, a **Figura 8** mostra a visualização de um volume analítico que consiste em um conjunto de cubos de diferentes dimensões, centrados no mesmo ponto. É possível verificar que a função de transferência ressalta corretamente apenas as faces dos cubos.

As figuras abaixo mostram resultados de volumes mais complexos:



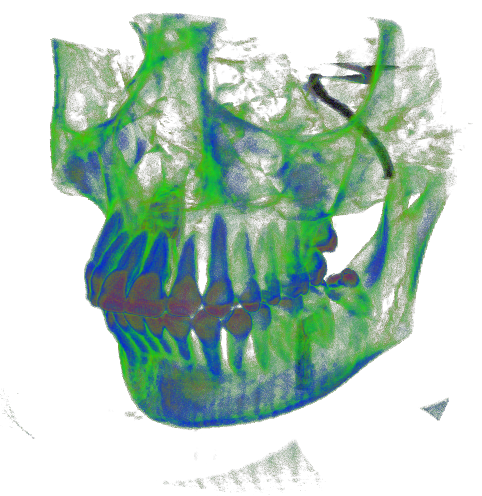
**Figura 9:** Visualização do volume de um bule (com uma lagosta dentro), utilizando uma função de transferência gerada automaticamente.



**Figura 10:** Visualização do volume de um pé, utilizando uma função de transferência gerada automaticamente.



**Figura 11:** Visualização do volume de uma lagosta, utilizando uma função de transferência gerada automaticamente.



**Figura 12:** Visualização do volume de um crânio, utilizando uma função de transferência gerada automaticamente.