PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Programa de Pós-Graduação em Engenharia Elétrica

Rosinei Soares de Figueiredo

# AVALIAÇÃO DO DESEMPENHO E DA QUALIDADE DO RESULTADO DO USO DE CUDA EM VISUALIZAÇÃO CIENTÍFICA VOLUMÉTRICA:

## Cálculo do uso do espaço tridimensional de vida de primatas como estudo de caso.

Belo Horizonte
2013

Rosinei Soares de Figueiredo

# AVALIAÇÃO DO DESEMPENHO E DA QUALIDADE DO RESULTADO DO USO DE CUDA EM VISUALIZAÇÃO CIENTÍFICA VOLUMÉTRICA:

## Cálculo do uso do espaço tridimensional de vida de primatas como estudo de caso.

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Dr. Carlos Augusto Paiva da Silva Martins.
Coorientadora: Dra. Flávia Magalhães Freitas Ferreira.

Belo Horizonte
2013

FICHA CATALOGRÁFICA
Elaborada pela Biblioteca da Pontifícia Universidade Católica de Minas Gerais

Figueiredo, Rosinei Soares de
F475a Avaliação do desempenho e da qualidade do resultado do uso de CUDA em visualização científica volumétrica: cálculo do uso do espaço tridimensional de vida de primatas como estudo de caso / Rosinei Soares de Figueiredo. Belo Horizonte, 2013.
70 f. : il.

Orientador: Carlos Augusto Paiva da Silva Martins
Coorientadora: Flávia Magalhães Freitas Ferreira
Dissertação (Mestrado) – Pontifícia Universidade Católica de Minas Gerais. Programa de Pós-Graduação em Engenharia Elétrica.

1. Visualização da informação. 2. Processamento paralelo (Computadores). 3. Modelos matemáticos. 4. Algoritmos de computador. 5. Linguagem de programação (Computadores). I. Martins, Carlos Augusto Paiva da Silva. II. Ferreira, Flávia Magalhães Freitas. III. Pontifícia Universidade Católica de Minas Gerais. Programa de Pós-Graduação em Engenharia Elétrica. IV. Título.

SIB PUC MINAS

CDU: 681.3.03

Rosinei Soares de Figueiredo

# AVALIAÇÃO DO DESEMPENHO E DA QUALIDADE DO RESULTADO DO USO DE CUDA EM VISUALIZAÇÃO CIENTÍFICA VOLUMÉTRICA:

## Cálculo do uso do espaço tridimensional de vida de primatas como estudo de caso.

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica.

_____

Dr. Carlos Augusto Paiva da Silva Martins (Orientador) – PUC Minas

_____

Dra. Flávia Magalhães Freitas Ferreira (Coorientadora) – PUC Minas

_____

Dr. Luiz Fabrício Wanderley Góes – PUC Minas

_____

Dr. Domingos da Costa Rodrigues – UFMG

Belo Horizonte, 11 de julho de 2013.

*Dedico este trabalho à minha família, aos meus amigos, e a todos que me estendem a mão, pois são eles o suporte da minha vida.*

## AGRADECIMENTOS

Agradeço primeiramente ao Senhor do Céu, da Terra, do Universo, pelos dons concedidos, pelas oportunidades, pelos caminhos que me permitiu.

Agradeço a minha mãe, em memória, e a todos os meus familiares, pelo apoio e suporte incondicional, sempre. À minha companheira, pela paciência e compreensão, pelo carinho e amor a mim dedicados.

Agradeço ao meu orientador Prof. Carlos Augusto, à minha coorientadora Profa. Flávia Freitas, aos colaboradores Prof. Robert Young e Marina Bonde, pelos ensinamentos, conselhos, sugestões e críticas que foram fundamentais para a realização deste trabalho.

Agradeço aos amigos do PPGEE, Alisson Rodrigo, Dayse Maria e Milene Barbosa, pelas sugestões e parcerias que em muito contribuíram para esta pesquisa.

Agradeço a todos do PPGEE, à Novaes, à Siqueira, ao Alan, aos colegas de turma e de laboratório, aos professores e a todos que participaram desta jornada.

Agradeço à CAPES pelo providencial apoio financeiro.

## RESUMO

Esta pesquisa realizou uma investigação exploratória acerca da dificuldade de se construir aplicações de Visualização Científica que apresentem resultados satisfatórios, e em tempo hábil, aos pesquisadores. Devido à amplitude do tema, a pesquisa o abordou de forma reduzida, tomando-lhe uma instância específica, através de um estudo de caso definido em conjunto com o Programa de Pós-Graduação em Zoologia dos Vertebrados da PUC Minas. O objetivo principal foi investigar o uso de GPUs programadas em CUDA como forma de viabilizar a execução de tais aplicações, que são geralmente vinculadas a contextos de grandes quantidades de dados e algoritmos complexos. Foram construídos nesta pesquisa: um modelo matemático para o problema especificado no estudo de caso, os algoritmos sequenciais e paralelos baseados no modelo matemático, a aplicação que implementa tais algoritmos e apresenta os resultados numéricos do processamento, bem como a apresentação visual simplificada dos resultados através de imagens. A aplicação construída foi experimentada em algumas plataformas paralelas baseadas em CPUs *multicore* e GPUs, tendo como foco principal as plataformas baseadas em GPU. Os resultados mostraram que o uso de computação paralela com GPU pode trazer ganhos significativos de desempenho e, ao mesmo tempo, consideráveis ganhos na qualidade dos resultados das aplicações de Visualização Científica.



Palavras-chave: Visualização Científica. Computação paralela. Unidade de Processamento Gráfico. CUDA C/C++.

## ABSTRACT

This research conducted an exploratory investigation about the difficulty of building applications for Scientific Visualization showing satisfactory results, and in a timely manner, to the researchers. Due to the breadth of the topic, the research addressed the reduced form, making it a specific instance, through a case study defined in conjunction with the Graduate Program in Zoology of Vertebrates at PUC Minas. The main objective was to investigate the use of GPUs in CUDA programmed as a way to enable the execution of such applications, which are usually linked to contexts of large amounts of data and complex algorithms. Were constructed in this research: a mathematical model for the problem specified in the case study, the sequential and parallel algorithms based on the mathematical model, the application that implements these algorithms and presents the numerical results of the processing as well as the visual presentation of results through simplified images. The application was built experienced in some parallel platforms based on multicore CPUs and GPUs, focusing mainly on GPU-based platforms. The results showed that the use of parallel computing GPU can provide significant performance gains at the same time, considerable gains in the quality of the results of scientific visualization applications.



Keywords: Scientific Visualization. Parallel computing. Graphic Processing Unit. CUDA C/C++.

## LISTA DE FIGURAS

**SUMÁRIO**

# 1 INTRODUÇÃO

Este capítulo contextualiza o campo de abordagem do trabalho, definindo também o problema a ser solucionado, a proposta de solução, os objetivos que compõem a proposta, as justificativas para a escolha do tema e o escopo da pesquisa. Também encontra-se detalhada aqui a organização do restante do documento.

## 1.1 Contextualização

Os modos de se fazer ciência estão mudando bastante no decorrer das últimas décadas. O avanço da tecnologia está permitindo pesquisas mais avançadas, que exploram objetos mais complexos, utilizando meios mais robustos. Nesse pacote tecnológico, a computação tem desempenhado papel muito importante, sendo considerada por alguns como um terceiro pilar da ciência, na forma da Ciência Computacional[1] (PITAC, 2005), e por outros, não como um novo pilar, mas como recurso fundamental para sustentar os pilares existentes, a Ciência Experimental e a Ciência Teórica (VARDI, 2010).

Não obstante a discussão terminológica, os autores evidenciam o envolvimento e a importância da computação na ciência. Cientistas e engenheiros fazem ciência computacional quando criam programas que modelam e executam os sistemas em estudo. Geralmente, esses sistemas apresentam grande complexidade advinda da sua própria lógica, do número de variáveis e dados envolvidos e suas inter-relações, e a computação vem sendo um fator fundamental para viabilizar e atribuir ganhos de qualidade às pesquisas.

A Ciência Computacional é formada pelos seguintes elementos: algoritmos, modelagem e simulação; Ciência da Computação e da Informação; e Infraestrutura de computação. Ela pode ser definida como a utilização de capacidades avançadas de computação para entender e resolver problemas complexos (PITAC, 2005).

Os computadores apresentam o potencial de modelar e simular fenômenos com mais rapidez e precisão do que o cérebro humano. Seu uso tem atingido muitas áreas de pesquisa, como estudos de genoma, de processos bioquímicos, da física de partículas, de cálculo de campo eletromagnético, de mudanças climáticas, de fenômenos epidemiológicos ou de

[1] *Computational science is a rapidly growing multidisciplinary field that uses advanced computing capabilities to understand and solve complex problems* (PITAC, 2005).

interações sociais com milhões de participantes (por exemplo, em comunidades digitais ou na Web) (SBC, 2006).

No entanto, alguns problemas colaterais são criados, uma vez que modelos muito complexos e volumes de dados cada vez maiores são gerados. Esses dados precisam ser analisados e trabalhados para uma efetiva extração de informação e conhecimento, criando a necessidade de meios rápidos e robustos no tratamento e visualização de dados.

Para visualização de grandes e complexas massas de dados, a comunidade científica tem se valido muito da Computação Gráfica. Gerar imagens que sintetizam uma grande quantidade de dados é de grande ajuda aos cientistas e engenheiros, pois isso facilita a compreensão global do fenômeno representado, a análise virtual do seu comportamento e a descoberta de conhecimentos de forma mais natural, quando comparado à análise de imensas quantidades de dados tabelados, sem nenhuma forma de abstração. A essa arte de transformar e selecionar grandes volumes de dados, quer sejam gerados por experimentos ou observações, quer por simulações numéricas, e transformá-los em imagens, dá-se o nome de Visualização Científica (MCCORMICK; DEFANTI; BROWN, 1987) (WIKIPEDIA, 2011).

O objetivo da Visualização Científica é facilitar a compreensão, a extração de informação e a construção de conhecimento acerca de grandes e complexos conjuntos de dados. A extração de informação refere-se à busca e apresentação de dados contextualizados e com algum significado para o observador. Já a construção de conhecimento baseia-se na interpretação da informação pelo observador, com base na sua experiência adquirida e em modelos.

A Visualização Científica é uma estratégia interessante para visualização de dados, ela tem sido muito estudada e discutida em grandes eventos e grupos de pesquisa. Por exemplo, o relatório PITAC (2005) indica, entre outras, a necessidade de se criar "*novos algoritmos e técnicas em Visualização Científica, para permitir capturar de forma visual a complexidade dos objetos modelados e de suas interações*".

Tomada como um processo, a Visualização Científica pressupõe algumas etapas, tais como a como aquisição de dados, a melhoria e a transformação desses dados, o mapeamento de visualização, a filtragem de dados, a renderização e a formação da imagem propriamente dita. Cada uma dessas etapas será melhor apresentada na seção 2.1 deste documento.

Um dos principais desafios relacionados a Visualização Científica é o alto custo computacional que as suas aplicações possuem. Pela natureza dos dados, geralmente muito complexos e massivos, as aplicações de visualização acabam por ter um tempo de resposta

computacional lento e uma precisão dos resultados (tamanho máximo da amostra processada e resolução da visualização, por exemplo) não muito satisfatória.

Assim, tanto para considerar a massa de dados em sua completude, quanto para dar um resultado mais preciso e com tempo de resposta satisfatório, as aplicações tornam-se consumidoras de grandes quantidades de recursos computacionais, principalmente a capacidade de processamento. Para suprir essa demanda, a literatura tem mostrado que a utilização de computação paralela é um caminho promissor, em amadurecimento, ainda em crescimento e evolução (DONGARRA, 2006) (KINDRATENKO; TRANCOSO, 2011). Não obstante, a diversidade de soluções e tecnologias para computação paralela, bem como as características particulares de cada problema, podem dificultar a decisão sobre qual caminho seguir, ou, até mesmo, induzir à escolha de uma estratégia que não seja a mais adequada.

Este trabalho abrange o processo de Visualização Científica com foco na transformação de dados. Isto porque, dadas as características do problema aqui abordado, o qual será detalhado mais à frente, esta etapa é a que possui a maior margem para novas contribuições.

Este trabalho foi executado no contexto da aplicação da computação paralela em Visualização Científica, com o objetivo de suprir demandas de visualização de áreas externas à computação, em ciências não exatas. É, portanto, uma pesquisa exploratória, que buscou apresentar como a Visualização Científica pode ajudar na construção de conhecimento científico e de como a computação paralela pode impulsionar esses ganhos. As principais contribuições estão na aplicação de ferramental computacional em áreas carentes deste, não sendo uma contribuição pura para a computação ou para o domínio da aplicação.

## 1.2 Problema motivador

Como já mencionado, a Visualização Científica está quase sempre relacionada a contextos de grandes quantidades de dados e algoritmos complexos, utilizados na geração, transformação e visualização desses dados. Assim, muitas vezes as aplicações construídas geram visualizações de baixa precisão e resolução, por considerarem apenas partes amostrais dos dados, e com tempos de resposta demorados, o que cria uma grande demanda por recursos computacionais quando da necessidade de se construir tais aplicações com níveis mais altos de qualidade e desempenho satisfatório (FOLEY et al., 1996).

O problema geral desta pesquisa pode, então, ser apresentado na forma do questionamento de como é possível viabilizar a construção e execução de aplicações de Visualização Científica com ganhos significativos de desempenho e qualidade dos resultados.

Por ser um tema de amplitude larga, a pesquisa o abordou de forma reduzida, tomando-lhe uma instância específica, através de um estudo de caso definido em conjunto com o Programa de Pós-Graduação em Zoologia dos Vertebrados (PPGZV) da PUC Minas. Tal estudo aborda o cálculo e a apresentação visual do espaço de vida utilizado por alguns grupos de primatas à paritr de amostras de localização geográfica feitas através da abservação destes animais. O estudo é detalhado na seção 4.1 deste documento.

## 1.3 Proposta de Solução

Para viabilizar a utilização de aplicações que demandem muito poder de processamento, as principais estratégias adotadas atualmente são baseadas em computação paralela (DONGARRA, 2006). Dentre essas estratégias, o uso de unidades de processamento gráfico (GPUs) como componentes de processamento de propósito geral, e sua programação em *Compute Unified Device Architecture* (CUDA), têm se destacado bastante, pois esses componentes têm apresentado uma capacidade de processamento em paralelo muito eficiente com uma boa relação custo-benefício, enquanto CUDA tem se mostrado uma plataforma fácil e eficiente para a programação de GPUs.

Baseado nisto, a hipótese deste trabalho foi de que o uso de computação gráfica com GPU poderia viabilizar a execução de aplicações de Visualização Científica. Propõe-se então usar GPUs, programando-as em CUDA, como forma de prover capacidade computacional para a aplicação construída, focando-se em melhorias no tempo de resposta e na qualidade das visualizações. A verificação da proposta foi feita utilizando cargas de trabalhos provenientes do estudo de caso definido conjuntamente com o PPGZV.

## 1.4 Objetivos e Metas

O objetivo principal deste trabalho é investigar o uso de GPUs com CUDA como forma de viabilizar a execução de aplicações de Visualização Científica vinculadas às demandas específicas do PPGZV. Como forma de alcançar o objetivo principal e adquirir conhecimentos teóricos e práticos para sua realização, considerou-se as demandas do PPGZV através de um estudo de caso, para o qual foram estabelecidos os seguintes objetivos:

- Formalizar o estudo de caso, com o detalhamento das suas especificidades;
- Modelar a aplicação de Visualização Científica relacionada ao estudo de caso;
- Estudar as características das arquiteturas de computação paralela utilizadas e a tecnologia CUDA;
- Construir e experimentar versões da aplicação a fim de explorar as possibilidades das arquiteturas paralelas utilizadas e encontrar meios para explorar melhor as suas potencialidades;
- Analisar os resultados dos experimentos do ponto de vista dos ganhos de desempenho e da qualidade dos resultados, considerando o atendimento às necessidades dos pesquisadores do PPGZV.

Algumas metas foram definidas com o propósito de estabelecer marcos a serem alcançados pela pesquisa. A principal delas é a definição de um conjunto de métricas de desempenho das arquiteturas utilizadas, quando da execução das diferentes cargas de trabalho, na forma dos programas paralelos construídos e executados nas mesmas.

Outras metas importantes são:

- A especificação do conjunto de características das cargas de trabalho geradas, que sejam importantes para os experimentos;
- A definição do conjunto de características arquiteturais importantes das arquiteturas paralelas utilizadas, que então devam ser observadas;
- As versões sequenciais e paralelas da aplicação para que sejam comparadas;
- O conjunto de resultados das experimentações e suas análises.

## 1.5 Justificativa

A computação é, de fato, valiosa para a ciência e para engenharia, não importando se ela é um terceiro pilar ou se é apenas uma sustentação para os dois pilares existentes. Fato é que com a computação, a ciência e a engenharia têm vencido novos desafios a cada dia, podendo ir além de suas antigas fronteiras (VARDI, 2010).

No contexto do crescente envolvimento da computação nas demais áreas, um campo de importância expressa em grandes relatórios como o PITAC (2006) e o SBC (2006) dos Estados Unidos e Brasil, respectivamente, é a Visualização Científica, objeto de pesquisa tanto para o desenvolvimento do modo como é utilizada pelas áreas de conhecimento, quanto para o desenvolvimento de seus próprios métodos, técnicas e ferramentas.

Nesse mesmo contexto, mas em esfera local, considerou-se a crescente demanda por aplicações envolvendo Visualização Científica, como o caso da demanda do PPGZV da PUC Minas. Projetos desenvolvidos no PPGZV precisam de ferramental para análise e avaliação de dados, sobretudo dados oriundos de observações. Nesse sentido o Prof. Dr. Carlos Augusto Paiva da Silva Martins, juntamente com o Prof. Dr. Robert Young, fecharam uma parceria para aplicação de conhecimentos de Matemática, Engenharia e Computação na área de Zoologia dos Vertebrados, ainda incipiente desses mecanismos.

Atualmente, a parceria tem realizado projetos referentes à análise da utilização da área de vida pelos animais. A área de vida, ou *home range*, de um animal é a área onde ele passa seu tempo, ou seja, a região que engloba todos os recursos que o animal necessita para sobreviver e reproduzir-se (BONDE, 2011). Nessas análises, uma necessidade importante é de que se chegue a uma representação da área de vida usada pelos animais. A representação deve ser sazonal, por grupos, ou por algum outro filtro definido e deve fornecer saídas numéricas e saídas visuais para descrição da área de vida utilizada. Muitas técnicas bidimensionais são usadas, mas percebeu-se que elas acabam perdendo grandes quantidades de informação e conhecimento, uma vez que a maioria dos animais utiliza uma terceira dimensão em seu ambiente de vida, representada pelo seu posicionamento vertical.

Com a parceria, algumas alternativas de análise da área de vida em três dimensões foram propostas, e uma delas é o estudo de caso deste trabalho. Trata-se da criação e implementação de uma técnica de análise de área de vida baseada na técnica *Kernel* (WORTON, 1989), uma das principais utilizadas para esse tipo de análise quando apenas duas dimensões são consideradas. Esta técnica baseia-se na densidade de utilização do espaço e por isso é capaz de separar regiões mais ou menos utilizadas e fornecer valores de utilização mais precisos do que as demais técnicas existentes, que somente consideram os contornos extremos das regiões utilizadas e apresentam valores superestimados. Não obstante, tal análise, que se trata de uma aplicação de Visualização Científica, só foi possível mediante a aplicação de técnicas de computação paralela.

Por fim, este foi o primeiro trabalho a abordar a área de Visualização Científica no âmbito do PPGEE, aplicando ferramental de Computação e de Engenharia em Ciências, na Biologia, e mais especificamente na Zoologia, sendo pioneiro na utilização da interdisciplinaridade, envolvendo outro programa de pós-graduação fora das áreas de ciências exatas.

## 1.6 Escopo

Compõe o escopo desta pesquisa a exploração da aplicação de computação paralela em Visualização Científica, tomando-se em um estudo de caso específico. Faz parte da pesquisa analisar o problema, propor e implementar uma solução de software e executá-la com cargas de trabalho provenientes do estudo de caso. Prevê-se também a análise das execuções da aplicação nas diferentes plataformas selecionadas, a fim de se avaliar, principalmente, o tempo de resposta (com os ganhos de desempenho) e a qualidade das visualizações (quanto ao tamanho da amostra utilizada).

O foco do trabalho foi a etapa de transformação dos dados, na qual foram aplicadas as técnicas de computação paralela. As demais etapas do processo de Visualização Científica foram abordadas com ênfase menor e de maneira mais simplificada.

Estão inclusas no escopo da pesquisa a definição e a implementação dos algoritmos necessários, bem como sua execução na arquitetura paralela selecionada, excluindo-se a análise minuciosa de linguagens de programação, compiladores, ambientes de desenvolvimento e demais tecnologias envolvidas, que serão definidas considerando-se pesquisa em literatura, disponibilidade e opiniões de especialistas.

Não faz parte do escopo deste trabalho uma avaliação sistemática das arquiteturas paralelas disponíveis para a seleção, o que será feito por meio de avaliações teóricas, disponibilidade e consultas a especialista em computação paralela. Exclui-se a responsabilidade de uma revisão teórica e/ou bibliográfica aprofundada tanto em Visualização Científica, quanto em computação paralela.

O objetivo principal é verificar a aplicação do paralelismo, utilizando GPUs com CUDA, como forma de permitir uma construção de conhecimento mais efetiva pelos cientistas e engenheiros, excluindo-se do escopo gerar, obrigatoriamente, contribuições científicas específicas para a área da computação ou para a área do domínio do problema abordado.

# 2 FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta os conceitos gerais necessários para a execução do trabalho.

## 2.1 Visualização Científica

Visualização Científica é uma área bastante ativa no que se refere à pesquisa, aprendizado e desenvolvimento. Trata-se do uso de imagens geradas por computador, que através de elementos geométricos dispostos topologicamente, potencializa o ganho de informação e entendimento sobre conjuntos de dados e suas relações (HAGEN et al., 2003). Está intimamente ligada a contextos onde grandes e complexas massas de dados de simulação de fenômenos físicos, de projetos de engenharia, de observação ou experimentação, precisam ser analisadas. Esta seção apresenta uma breve história da Visualização Científica, com ênfase em visualização volumétrica, devido ao escopo do trabalho.

A referida área é relativamente nova. Oficialmente, Visualização Científica surgiu na forma do termo *Visualization in Scientific Computing* – Visualização em Computação Científica, no relatório do *National Science Foundation's Advisory Panel on Graphics, Image Processing, and Workstations*, em 1987 (MCCORMICK et al., 1987).

Neste relatório, McCormick e outros (1987) expressam a importância da computação, como uma área emergente entre a teoria e a experimentação. Fala-se sobre o movimento em que a computação surge com potencial de promover ganhos significativos para ciência e engenharia, de forma distinta em cada caso, mas com muitos aspectos comuns, principalmente a utilização da visualização para apresentação de dados simulados ou coletados. Alguns exemplos de domínios de aplicação são citados: modelagem molecular, imagens médicas, estrutura e função do cérebro, matemática, geociência, exploração do espaço e astrofísica, em ciência; e dinâmica de fluidos e elementos finitos, em engenharia.

Com a utilização de aplicativos computacionais, muitos novos experimentos e projetos tornaram-se possíveis. No entanto, montanhas de dados começaram a se formar em meio a essas novas aplicações, tornando árdua e impraticável sua análise pelos cientistas e engenheiros. Nesse contexto, a Visualização Científica surgiu com o objetivo de ajudá-los na tarefa de interpretar os resultados de seus trabalhos.

Desde sua proposta inicial, é explícita a multidisciplinaridade como um forte aspecto da Visualização Científica. Componentes de *hardware* e *software*, disciplinas como computação gráfica, processamento de imagem e visão computacional, conjuntos de símbolos

e formatos de imagens, entre outros, formam um conjunto de elementos que se interconectam e se interagem, constituindo um sistema maior que provê as funcionalidades necessárias. Alguns trabalhos encontrados na literatura mostram o quanto a Visualização Científica evoluiu desde que foi criada, bem como as tendências da área para o futuro. Entre eles, destacam-se os trabalhos de Max (2005), Hansen e Johnson (2005) e Fuchs e Hauser (2009).

Em Max (2005), é apresentada uma análise dos progressos alcançados pela Visualização Científica nos últimos vinte anos anteriores ao ano de publicação do trabalho, orientando seu estudo para visualização em moléculas, campos escalares (conjunto de dados escalares, números puros representando valores únicos, com sinal) e campos vetoriais (conjunto de dados vetoriais, com informações de magnitude, sinal e orientação).

Em visualização molecular, o artigo aborda dois tipos de progressos: aqueles que são especificamente químicos e aqueles que são mais instrumentais. Sobre avanços químicos é citado o desenvolvimento na modelagem de estruturas moleculares, com a criação de novos modelos que agregam os modelos já existentes, aprimorando-os com a adição de outros elementos. Já sobre avanços instrumentais, são citados os avanços na modelagem apoiada por computador, além de avanços nas tecnologias de imagem e vídeo, que permitem manipulações sobre os modelos construídos e gravação de seus comportamentos próprios em formato de vídeo eletrônico.

Em campos escalares, são abordados alguns avanços em renderização volumétrica. Uma das grandes novidades foi o surgimento dos primeiros trabalhos abordando a representação de volume com semi-transparência, construída por meio da atribuição de cores transparentes e diferenciadas para cada região do volume, considerando sua densidade, de grande auxílio quando se precisa compreender a estrutura interna dos objetos. Outros progressos mais tecnológicos também foram relatados. Neles se incluem o uso de *hardware* de propósito geral (CPUs) e específico (GPUs) para implementação de determinadas atividades da visualização, como a criação de texturas 3D, o mapeamento do ambiente tridimensional para a imagem bidimensional, a renderização, entre outros. Por fim, o autor registra a preocupação com a necessidade de se desenvolver métodos inteligentes para classificação da importância das regiões de volume, juntamente com técnicas para dar ênfase a regiões durante a visualização, a fim de minimizar grandes perdas de informação durante o processo.

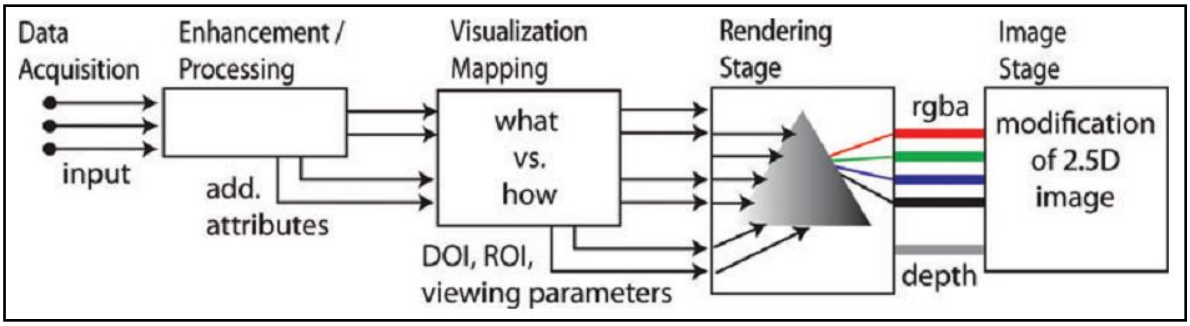
Ainda sobre campos escalares, são também abordados alguns avanços acerca de contorno das superfícies. O principal progresso relatado refere-se aos métodos utilizados para

criação de contornos de volume, principalmente com a mudança das técnicas baseadas em sombreamento para as técnicas baseadas em triangulação (MAX, 2005).

Por fim, ao tratar sobre campos vetoriais, o autor aborda os progressos observados em visualização de fluxos. Estes progressos são, principalmente, em torno da utilização de técnicas inspiradas nas experimentações da dinâmica de fluidos, o que gerou ganhos significativos quando comparada com as técnicas baseadas no desenho de setas, principalmente em grades tridimensionais.

Alguns outros trabalhos abordam elementos mais relacionados ao processo de se fazer Visualização Científica, destacando-se o trabalho de Fuchs e Hauser (2009). Neste trabalho é expressa a necessidade de se conhecer bem as técnicas de visualização existentes, pois as visualizações mais efetivas só podem ser obtidas quando da aplicação da técnica ou do conjunto combinado de técnicas mais apropriadas. No entanto, para que uma técnica seja bem utilizada, também é importante saber em qual momento da visualização ela pode ser aplicada. Com este objetivo, em Fuchs e Hauser (2009) é apresentado um *pipeline* de visualização. Trata-se de um modelo abstrato que "descreve como representações visuais de dados podem ser obtidas seguindo uma sequência de passos bem definidos", como pode ser observado na Figura 1, que segue:

**Figura 1: Pipeline de Visualização**




**Fonte: Fuchs e Hauser (2009)**

O estágio de **Aquisição de Dados** trata da medição ou geração dos dados científicos, especificamente; o estágio de **Melhoria e Transformação** é responsável por entregar os dados de forma padronizada e distribuível, aplicando para isto algumas técnicas de transformação, se necessárias; o estágio de **Mapeamento de Visualização**, também chamado de Filtragem de Dados, tem a responsabilidade de remover ou agrupar dados irrelevantes e, a partir do conjunto de dados final e das informações preliminares, mapeá-los para as representações visuais; o estágio de **Renderização** é responsável por traduzir em imagens as

informações contidas nos dados, já mapeadas em representações específicas pela etapa anterior; por fim, o estágio denominado **Imagem** tem a função de permitir a manipulação da imagem renderizada para se chegar a uma visualização final.

Para cada um dos estágios apresentados na Figura 1 existem métodos e técnicas associados e portanto faz parte de cada projeto definir qual ou quais serão os métodos e técnicas utilizados em cada estágio. Além disso, nem sempre um projeto de visualização contém todos os estágios apresentados, além do que sua organização pode variar, eles podem ser reorganizados e recombinados, dispostas em paralelo ou em sequência, dependendo das características particulares de cada projeto, envolvendo o problema e as tecnologias utilizadas.

Esta pesquisa abordará os blocos de "melhoria e transformação", "mapeamento de visualização" e "renderização". O estágio de "melhoria e transformação" refere-se à aplicação do método de cálculo de utilização da área de vida dos animais pelo método proposto e compõe o objeto principal de investigação. O estágio de "mapeamento de visualização" refere-se à tradução dos dados resultantes da transformação em informações de cor e transparência. Por fim, o processo de "renderização" trata da geração de imagens que representem os dados processados. As duas últimas etapas não constituem o foco principal desta pesquisa e foram abordadas de forma simplificada.

### *2.1.1 Visualização Volumétrica*

Apresenta-se a seguir uma visão de localidade e uma definição geral sobre Renderização Volumétrica, com base nas seguintes referências: Hansen e Johnson (2005), Foley e outros (1996), McCormick; DeFanti e Brown (1987) e Paiva e outros (2011).

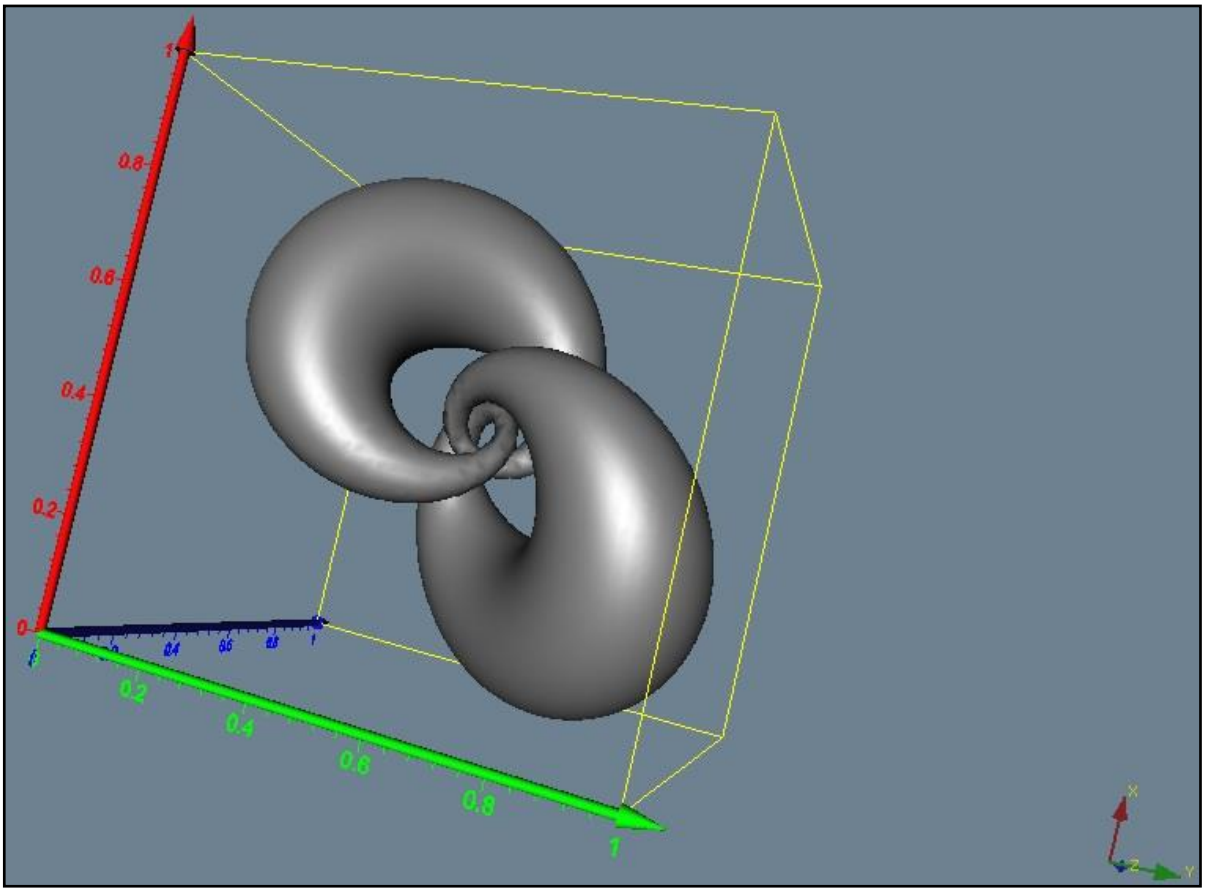
Nesta pesquisa serão trabalhados dados escalares coletados em ambientes tridimensionais. Assim, pode-se dizer que serão processados dados volumétricos escalares para construção das visualizações. Dessa forma, as visualizações construídas encaixam-se na categoria de visualização denominada visualização volumétrica.

A visualização volumétrica é o conjunto de técnicas utilizadas para visualização de dados volumétricos, escalares ou não, associados a regiões de volume, para facilitar sua compreensão por meio da visualização do seu interior e da exploração da sua estrutura (MCCORMICK et al., 1987).

Em visualização, o processo de transformar um ou mais modelos em uma imagem é chamado de renderização. Duas vertentes bem distintas de técnicas ou algoritmos podem ser observadas para esse propósito. A primeira é composta pelos algoritmos baseados em

contorno, os quais buscam extrair representações de superfície para pontos que assumem valores iguais (isossuperfícies, para dados em três dimensões), para então renderizar somente essas superfícies, como no exemplo da Figura 2. A segunda vertente é composta pelos algoritmos que buscam renderizar os dados diretamente (Renderização Volumétrica, para dados em três dimensões), com gráficos e imagens interativos compostos de primitivas geométricas, como no exemplo da Figura 3. Essa separação dos algoritmos de visualização volumétrica em algoritmos de extração de isossuperfície e algoritmos de renderização volumétrica pode ser analisada com mais detalhes em McCormick; DeFanti e Brown (1987), Elvins (1992), Foley e outros (1996) e Paiva e outros (2011).

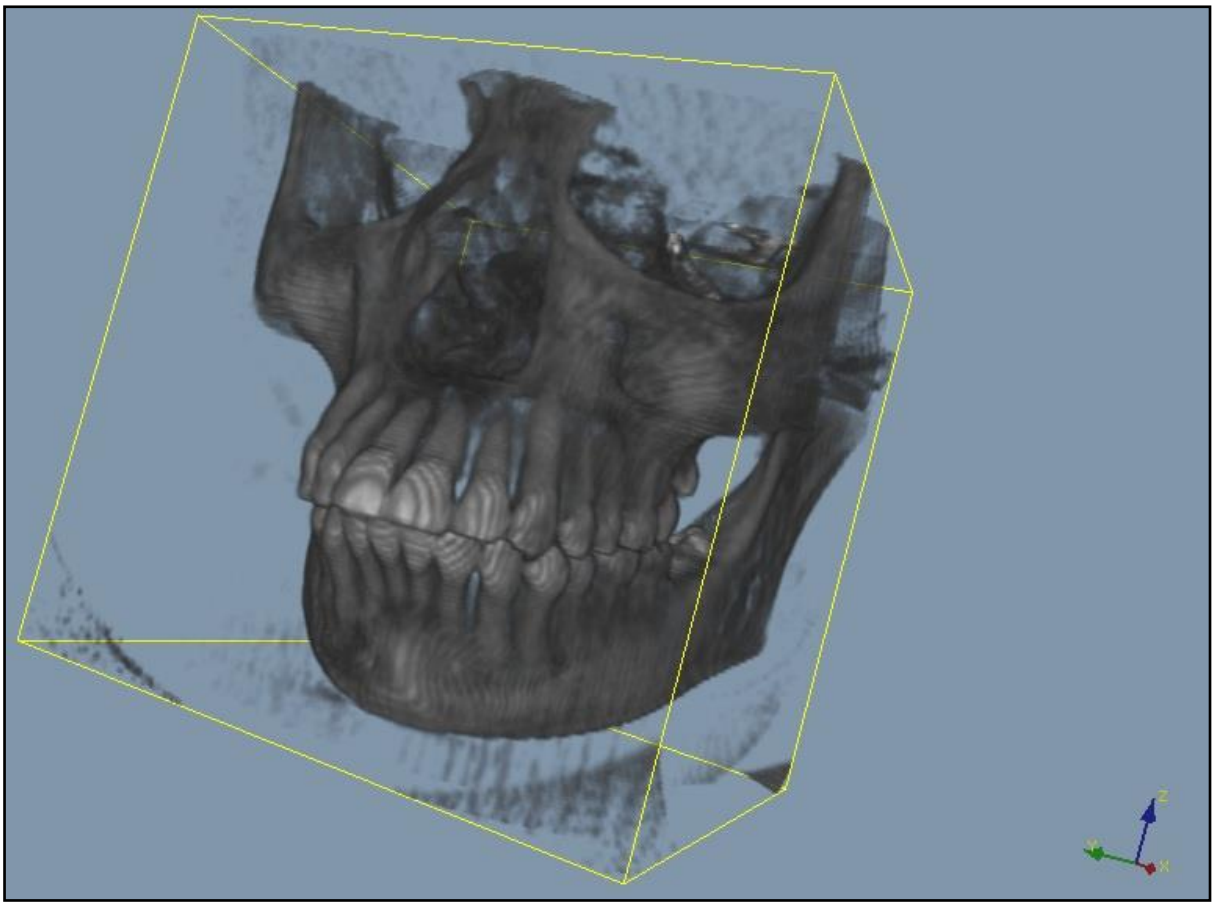
**Figura 2: Exemplo de Isossuperfície**




**Fonte: Exemplo retirado do programa de visualização Voxler**

Pela Figura 2 é possível notar que a imagem foi gerada à partir da percepção das superfícies do objeto apresentado, característica fundamental dos algoritmos de de extração de isosuperfície.

**Figura 3: Exemplo de Renderização Volumétrica**



**Fonte: Exemplo retirado do programa de visualização Voxler**

Por outro lado, na Figura 3, é possível perceber que a imagem apresenta as estruturas internas do objeto ao invés de somente suas superfícies, o que caracteriza os algoritmos de renderização volumétrica.

A segunda vertente possui a vantagem de evitar as perdas de informação na apresentação dos dados, tal qual ocorrem na primeira vertente, quando da simples renderização dos contornos extraídos, que apenas cria formas geométricas que representam todo o conjunto de dados, perdendo suas características internas. Por outro lado, a complexidade dos algoritmos e o custo computacional das operações da segunda vertente são muito grandes, exigindo a utilização de muitos métodos de otimização e a execução em *hardware* dedicado (GPU, por exemplo) ou outras estratégias de computação de alto desempenho.

A segunda vertente, renderização volumétrica, será explorada neste trabalho, utilizando-se a computação paralela em *hardware* dedicado (GPU) como meio de prover a capacidade computacional necessária.

## 2.2 Renderização Volumétrica

A Renderização Volumétrica, também chamada de Renderização Direta ou ainda Visualização Direta de Volume, refere-se à apresentação visual do volume como um todo. Cada célula ou elemento do volume é representado através de um *voxel* (*volume element* – elemento de volume) e esses elementos são processados, projetados diretamente em pixels (*picture element* – elemento de imagem), ou armazenados como uma imagem (ELVINS, 1992) (FOLEY et al., 1996) (MANSSOUR; FREITAS, 2002).

Dois elementos principais devem ser considerados em renderização direta: a Função de Transferência e o mecanismo de visita aos *voxels* [Manssour e Freitas 2002]. A Função de transferência corresponde à função que faz o mapeamento dos valores dos *voxels* (densidade ocupacional, por exemplo) para propriedades visuais, tais como cor e opacidade. Já o mecanismo de visita é o processo responsável pela visita dos *voxels* para definição da contribuição de cada um deles aos *pixels* da imagem resultante. Os principais algoritmos são *Splatting* (WESTOVER, 1989) (WESTOVER, 1990), *Shear-Warp* (LACROUTE; LEVOY, 1994), *Shell Rendering* (UDUPA; ODHNER, 1993), *Cell-Projection* (WILHELMS; VAN GELDER, 1991) e *V-Buffer* (UPSON; KEELER, 1988) e *Ray Tracing* (AZEVEDO; CONCI, 2003) (LEVOY, 1988) (LEVOY, 1990) (YAGEL; COHEN; KAUFMAN, 1992).

## 2.3 Computação Paralela

Etimologicamente, computação é o ato ou efeito de computar alguma coisa (FERREIRA, 1999), ou seja, calcular, processar alguma entrada e gerar alguma saída. Trata-se da busca pela solução de um problema aplicando-se algum processamento sobre entradas que representam o problema, e gerando-se saídas que compõem a solução para o problema. Diferente do que se observa hoje, a computação existe muito antes das máquinas eletrônicas, sendo feita com caneta e papel, giz, mentalmente, ou mesmo com auxílio de equipamentos mecânicos (WOOD, 1987) (JOHN, 1998). Utiliza-se aqui, para efeito de simplificação, o termo computação como somente aquela feita por computadores eletrônicos.

Essa computação eletrônica pode ser feita por computadores de vários tipos. A forma mais comum de classificar os tipos de computadores é através da Taxonomia de Flynn (FLYNN, 1966) (FLYNN, 1972), que definiu as seguintes categorias:

- Máquinas SISD (*Single instruction stream, single data stream* - Fluxo de única instrução, único dado): uma única sequência de instruções processando dados armazenados em uma única memória;
- Máquinas SIMD (*Single instruction stream, multiple data stream* - Fluxo de única instrução, múltiplos dados): uma única instrução de máquina coordena a execução de um conjunto de elementos de processamento, cada um com uma memória a ele associada. Assim, a mesma instrução é executada simultaneamente em diferentes conjuntos de dados;
- Máquinas MISD (*Multiple instruction stream, single data stream* - Fluxo de múltiplas instruções, único dado): um conjunto de processadores executando simultaneamente sequências diferentes de instruções em uma única sequência de dados;
- Máquinas MIMD (*Multiple instruction stream, multiple data stream* - Fluxo de múltiplas instruções, múltiplos dados): um conjunto de processadores executando simultaneamente sequências diferentes de instruções em múltiplos conjuntos de dados.

Segundo Stallings (2010), a Computação Paralela acontece em mais de uma das categorias propostas por Flynn, a saber, em:

- Máquinas SIMD, representada pelos processadores vetoriais e pelos processadores matriciais, onde o paralelismo acontece somente com os dados;
- Máquinas MISD, com múltiplos conjuntos de instruções executando sobre um mesmo conjunto de dados, embora, segundo muitos autores, não exista nenhuma máquina construída com este paradigma de processamento;
- Máquinas MIMD, representada pelos *clusters*, pelas máquinas *Non-Uniform Memory Access* (NUMA) e pelos processadores S*ymmetric Multiprocessing* (SMP), os processadores *multicore*, por exemplo.

De um ponto de vista mais abstrato, a computação paralela é definida como um processo bem definido. Inicialmente, o especialista da área da aplicação, cientista ou engenheiro, define um problema numérico ou não numérico que representa as suas demandas. Nesse ponto, o cientista da computação projeta um algoritmo paralelo para a solução do problema definido e implementa tal algoritmo utilizando uma linguagem de programação sobre um computador que possui componentes de hardware e software para permitir a execução de programas paralelos. Então, o especialista da área do problema executa o

programa para encontrar as soluções do problema definido. Este é o processo padrão, embora muitas vezes ele não seja tão rigidamente obedecido, frequentemente ocorrendo a construção de programas sequenciais para algoritmos naturalmente paralelos e, ainda, a execução de programas paralelos em computadores sequenciais e vice versa.

A computação paralela é, portanto, composta por: problemas de aplicação com características paralelas como motivadores; os algoritmos paralelos como base teórica; a programação paralela com forma de fornecer softwares com suporte necessário; e computadores paralelos, as plataformas de hardwares e softwares básicos necessárias à execução dos softwares paralelos (CHEN et al., 2009).

## 2.4 Plataformas paralelas atuais

Os computadores paralelos de destaque atualmente são construídos, principalmente, sobre uma ou mais das seguintes tecnologias: CPU *Multicore* (DEBES, 2003), GPU (MACEDONIA, 2003), APU (AMD, 2011) e FPGA (ROSE et al., 1990). *Multicore*, ou Multinúcleo, é um processador com dois ou mais núcleos de processamento em um único chip; *Graphics Processing Unit* (GPU) é uma unidade de processamento especializada em processamento gráfico; *Accelerated Processing Unit* (APU) é a fusão de uma CPU com uma GPU em um mesmo chip; *Field-Programmable Gate Array* (FPGA) é um tipo de hardware reconfigurável, que pode ser programado conforme a necessidade.

Neste trabalho foram utilizadas plataformas baseadas em CPU *multicore* e plataformas baseadas em GPU, sendo as últimas o foco principal da pesquisa.

### *2.4.1 Multicore*

Um componente de processamento *Multicore* é um microprocessador – *Central Process Unit* (CPU) – com dois ou mais núcleos de processamento em um único chip. O *multicore* começou a ser construído como alternativa à dificuldade em se continuar promovendo ganhos de desempenho utilizando os *single cores*, processadores de um único núcleo, principalmente pelas dificuldades em refrigerar os componentes de silício manipulados em escala nanométrica. Com os *multicores*, ao se incorporarem os vários núcleos de processamento em um único encapsulamento, torna-se possível a execução paralela de múltiplas *threads*, com possibilidades de ganhos significativos de desempenho e uso mais eficiente de energia (DEBES, 2003) (INTEL, 2011) (WIKIPEDIA, 2011B).

### *2.4.2 GPU*

Uma *Graphics Processing Unit* (GPU) é um microprocessador especializado em processar gráficos e geralmente está acoplado em uma placa de vídeo, mas também pode ser acoplado diretamente em placas mães, em algumas versões mais simplificadas (MACEDONIA, 2003).

Uma GPU traz implementadas em hardware as funções de processamento gráfico que consomem recursos computacionais em grande quantidade, tais como mapeamento de texturas, iluminação e transformação de vértices, entre outras operações, o que deixa o processador central livre para executar as demais tarefas do computador.

Investimentos em GPUs fizeram a tecnologia evoluir bastante. Hoje é possível encontrar GPUs com quantidade enorme de núcleos de processamento, com baixo consumo de energia e com níveis muito avançados de desempenho computacional.

Toda essa capacidade disponível acabou despertando o interesse da comunidade em expandir a utilização das GPUs, utilizando-as também para processamento geral no computador, criando o conceito de *General-Purpose Computing on Graphics Processing Units* (GPGPU), que pode ser traduzido para Computação de Propósito Geral em Unidades de Procesamento Gráfico, tecnologia com a qual o potencial das unidades de processamento gráfico é também explorado para processamento geral (MANDL; BORDOLOI, 2011) (WIKIPEDIA, 2011c).

### *2.4.3 Programação CUDA*

A crescente demanda por aplicações gráficas de alta definição e de tempo real tem impulsionado uma expressiva evolução nas GPUs programáveis. Constantemente são lançados novos modelos e arquiteturas e, cada vez mais, as GPUs têm apresentado mais capacidade de paralelização de aplicações, execução de múltiplas *threads* e melhorias no acesso à memória.

Uma das características observadas é a grande diferença entre as GPUs e CPUs no que se refere à quantidade de operações com ponto flutuante que elas podem processar por segundo. Nesse quesito, as GPUs demonstram capacidades muito superiores que a capacidade das CPUs, isso porque as primeiras são especializadas em computação intensiva e altamente paralela, para satisfazer principalmente as demandas das aplicações gráficas. Além disso, são

projetadas para enfatizar mais o processamento de dados do que o controle de fluxo ou de *cache*, como nas CPUs.

Para possibilitar a utilização de GPUs como componente de processamento geral, algumas plataformas de programação específicas para esse fim foram desenvolvidas, entre elas a plataforma CUDA, que foi abordada nesta pesquisa.

O termo CUDA (*Compute Unified Device Architecture*) refere-se a uma plataforma de computação paralela e a um modelo de programação criados pela empresa NVIDIA, em parceria com as empresas Adobe, ANSYS, Autodesk, MathWorks e Wolfram Research. Como plataforma, CUDA provê uma série de ferramentas que possibilitam aos desenvolvedores de *software* criarem aplicações que aproveitem o poder de processamento oferecido pelas GPUs da NVIDIA. Já como modelo de programação, CUDA provê um conjunto de modelos e indicações necessários para criação de programas que utilizem recursos de processamento de GPUs (NVIDIA, 2012a).

Existem muitas GPUs habilitadas para CUDA disponíveis no mercado atualmente, o que tem permitido avanços significativos de desempenho computacional no aproveitamento dos recursos dessas GPUs. Com isso, novas aplicações estão permitindo avanços em diversas áreas de conhecimento científico, como identificação de placas ocultas em artérias, análise de fluxo de tráfego aéreo, visualização de moléculas em escala nanométrica, entre outras (NVIDIA, 2012a).

Além de possibilitar ganhos significativos de desempenho, CUDA ainda tem a vantagem de diminuir a curva de aprendizado de programação paralela. Com CUDA, é possível ao programador enviar código em C, C++ e Fortran diretamente à GPU, sem precisar usar uma linguagem de compilação. Assim, os programas executam as partes *sequenciais* de suas cargas de trabalho na CPU – que geralmente é otimizada para desempenho com um único segmento (*thread*) – ao mesmo tempo em que aceleram o *processamento em paralelo* através da GPU (NVIDIA, 2012a).

A linguagem de programação utilizada nesta pesquisa foi a CUDA C, uma extensão da linguagem C, com algumas restrições necessárias à execução em GPUs. O trechos de códigos escritos em CUDA C são chamados *kernels*. Vale lembrar que esta nomenclatura não possui nenhuma relação com o método de cálculo implementado, que coincidentemente se chama Kernel. *Kernels*, neste contexto, são funções escritas em CUDA C que, quando invocadas, executam em uma GPU, várias vezes em paralelo, por diferentes CUDA *threads*, diferentes das funções convencionais escritas em C que executam em GPU, somente uma vez.

Um *kernel* é definido utilizando-se a declaração `__global__` na assinatura da função. A quantidade de *threads* que executa tal *kernel* é definida através de uma sintaxe específica de configuração de execução, conforme exemplo na Figura 4 a seguir.

**Figura 4: Exemplo de declaração e invocação de *kernel***

```
// Kernel definition
__global__ void VecAdd(float* A, float* B, float* C)
{
    int i = threadIdx.x;
    C[i] = A[i] + B[i];
}

int main()
{
    ...
    // Kernel invocation with N threads
    VecAdd<<<1, N>>>(A, B, C);
    ...
}
```

**Fonte: NVIDEA CUDA (2011)**

Como pode ser observado no exemplo acima, a invocação da função *kernel* possui um elemento adicional, uma seção limitada pelos sinais <<< e >>>. Nessa seção são definidos o número de threads que executarão a função *kernel* e a forma como elas serão organizadas, o que poderá ser melhor entendido mais à frente.

Dois parâmetros devem ser informados entre os operadores <<< e >>>. O primeiro é à quantidade de blocos de *threads* a ser utilizada na execução e o segundo é quantidade de *threads* dentro de cada bloco.

Cada *thread* em execução recebe um identificador único, acessível no escopo do *kernel*, através da variável `threadIdx`. Esta variável é, porém, um vetor com três dimensões, o que permite ao desenvolvedor rearranjar a distribuição dos *threads* da forma mais adequada ao contexto da sua aplicação, utilizando vetores, matrizes ou volumes.

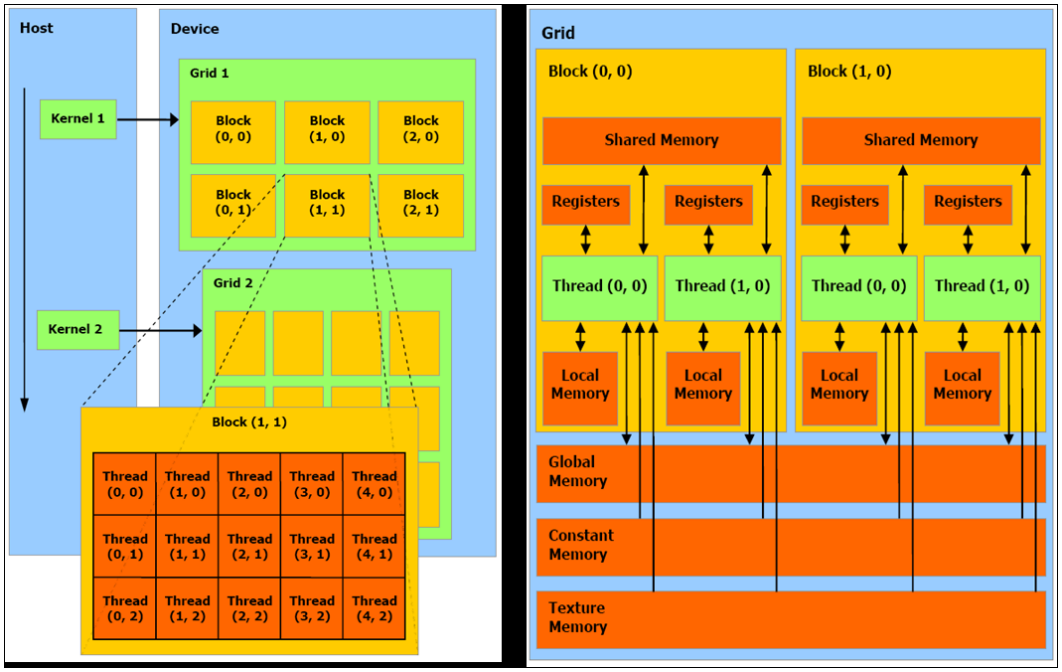
Com as possíveis organizações dos *threads*, o desenvolvedor pode escolher manipular um vetor de *threads* (identificadas pelo atributo x do `threadIdx`, acessível através de `threadIdx.x`); uma matriz bidimensional de *threads* (identificadas pelos atributos x e y do `threadIdx`, acessíveis através de `threadIdx.x` e `threadIdx.y`) ou uma matriz tridimensional de *threads* (identificadas pelos atributos x, y e z do `threadIdx`, acessíveis através de `threadIdx.x`, `threadIdx.y` e `threadIdx.z`).

Algumas vezes, mesmo com a utilização de uma estrutura bidimensional ou tridimensional para a organização da *threads*, torna-se necessária a sua identificação através de um identificador único. Tal identificador pode ser adquirido da seguinte forma: para blocos unidimensionais, o identificar é o valor x contido no `threadIdx`; para blocos bidimensionais o identificador é dado por (x + y Dx), onde Dx é a dimensão de x dos blocos utilizados; já para os blocos tridimensionais, o identificador é dado por (x + y Dx + z Dx Dy), onde Dx e Dy são, respectivamente, a dimensão x e a dimensão y dos blocos utilizados.

A quantidade de *threads* por bloco é limitada e nas GPUs atuais cada bloco pode possuir, geralmente, no máximo 1024 *threads*. Para contornar essa limitação, a variável que indica a quantidade de blocos também é um vetor com três dimensões, assim como a variável que indica a quantidade de *threads*. Dessa forma, as aplicações podem trabalhar com quantidades maiores de *threads*, sem a necessidade de ficarem restritas à quantidade máxima por bloco. O total de blocos a serem utilizados é definido, geralmente, considerando-se a quantidade de dados a serem processados.

As funções *kernels* podem acessar espaços de memórias predefinidos para manipular seus dados, conforme a hierarquia apresentada na Figura 5.

**Figura 5: Hierarquia de memória e de *threads* em CUDA**




**Fonte: Song (2013)**

Pela hierarquia de memória e *threads* apresentada na Figura 5, percebe-se que cada *thread* possui um espaço de memória privado; cada bloco possui um espaço de memória compartilhado por todas as *threads* que o compõem; todas as threads possuem acesso a um espaço de memória global e, além disso, todas as *threads* possuem acesso a dois outros espaços de memória, a memória de constantes e a memória de texturas.

# 3 TRABALHOS RELACIONADOS

A computação paralela vem sendo utilizada como apoio à pesquisa em diversas áreas de conhecimento. Em Visualização Científica, ela é muito importante como meio de suprir a demanda computacional de aplicações que possuem algoritmos complexos trabalhando conjuntos de dados grandes e complexos. Encontram-se na literatura muitos trabalhos envolvidos nesse caminho e alguns deles foram selecionados para servir como guia e/ou objeto de comparação para esta pesquisa. Entre eles, citam-se Westermann (1995), (Van Pelt; Vilanova e Van de Wetering (2010), Howison; Bethel e Childs (2011) e Monoley e outros (2011), que são trabalhos relacionados à aplicação de ferramental computacional em áreas de ciências, e o trabalho de Bonde (2011), que é um trabalho relacionado à área de aplicação do presente trabalho e que possui objeto de pesquisa afim.

O trabalho de Westermann (1995) aponta experiências mais antigas relacionadas à utilização de computação paralela para renderização volumétrica com *Ray Tracing*. A abordagem utilizada parte do princípio da divisão do volume a ser renderizado entre nodos de processamento. São avaliadas cargas de trabalho de tamanhos distintos e *clusters* com diferentes configurações. Os resultados mostraram ganhos de desempenho significativos para algumas combinações específicas de configuração de memória e nodos de processamento.

Em Van Pelt; Vilanova e Van de Wetering (2010) é apresentada a aplicação do paradigma de GPGPU para renderização volumétrica de partículas. Foi desenvolvido um *framework* que pode ser utilizado em várias aplicações de visualização. Todos os algoritmos são voltados para execução em GPU, utilizando-se C++ como linguagem de programação, juntamente com a API OpenGL 3D combinada com a linguagem de sombreamento GL (GLSL). São discutidas implicações arquiteturais que afetam o desempenho, como a latência e o tamanho de memória, e as características da carga de trabalho. Foram utilizados uma GPU e um processador *multicore* específicos e os resultados mostraram que, nesse contexto, a utilização de GPGPU promove ganhos significativos de desempenho em relação ao uso do *multicore*, sobretudo no processamento das tarefas básicas fornecidas pela API do *framework*.

Em Howison; Bethel e Childs (2011) é feita uma comparação entre arquiteturas baseadas em *multicore*, arquiteturas baseadas em GPU e arquiteturas baseadas em APU, também chamadas de arquiteturas híbridas, que usam conjuntamente as duas anteriores, aplicando-as em renderização volumétrica. Os resultados mostram que, para o caso verificado, as arquiteturas híbridas apresentam vantagens sobre as demais, pois permitem explorar o melhor de cada uma das arquiteturas componentes, o que reduz a quantidade de

comunicação entre cada núcleo de processamento e o tamanho dos dados comunicados, tornando a execução mais rápida. No entanto, a melhor alternativa depende da disponibilidade de arquiteturas paralelas e do problema a ser estudado, características da sua carga de trabalho, tais como tamanho e o tipo do conjunto de dados, a necessidade de comunicação entre os núcleos de processamento, entre outros, os quais devem ser avaliados em cada caso.

Em Monoley e outros (2011), é feita uma análise do uso de GPU para renderização paralela de volume, comparando-se abordagens em que as ordenações de dados são feitas previamente (proposta dos autores) com as abordagens em que as ordenações de dados são feitas posteriormente (disponíveis na literatura). Os resultados mostram que as abordagens com ordenação prévia, aliadas ao uso da computação paralela com GPU, podem ser a melhor opção em alguns casos, dependendo das características da carga de trabalho, como localização e tipo da distribuição dos dados.

O trabalho de Bonde (2011) apresenta um estudo sobre o uso da área de vida pelos primatas *Calicebus Nigrifons*, comumente conhecidos como Guigós. O escopo do projeto abrangeu a coleta presencial de dados sobre o comportamento e localização dos animais em seu *habitat* [2] natural; a análise estatística das dimensões coletadas e a análise quantitativa da área de vida, que envolve o cálculo da área e volume e sua apresentação visual.

No que toca ao cálculo da área de vida puramente, foram utilizados alguns métodos específicos, a saber o Mínimo Polígono Convexo (MPC) e o Kernel. O MPC (HAYNE, 1949) é um método clássico e consiste em unir os pontos mais extremos da distribuição das localizações, de forma a fechar o menor polígono possível sem admitir concavidades. Já o Kernel é um método que fornece informações sobre a intensidade de uso do espaço. Nesta estimativa, a área de vida é definida como a menor região que contém uma dada proporção das localizações do indivíduo, ou seja, as áreas em que a probabilidade de se encontrar esse indivíduo seja diferente de zero.

No entanto, a pesquisa de Bonde (2011) vislumbrou outros objetivos: o de desenvolver uma análise da área de vida baseada no uso do espaço em três dimensões (latitude, longitude e altura em relação ao solo), no intuito de fornecer uma análise com mais acurácia, uma vez que os animais estudados, assim como muito outros, utilizam o espaço em três dimensões. Para tanto, o volume de vida dos grupos foi calculado de duas maneiras: a primeira forma utilizando a AM (Altura Média), em que a média das alturas dos indivíduos estudados foi

[2] *Habitat* é o ambiente onde determinada espécie vive, incluindo recursos para sua alimentação e condições para reprodução (BONDE, 2011).

calculada e posteriormente multiplicada pela área de vida. O segundo cálculo foi utilizando MPC3 (Mínimo Poliedro Convexo), um método similar ao MPC, estendido para três dimensões. O método MPC3 foi um contribuição específica da pesquisa de Bonde (2011), sendo desenvolvido durante a sua atuação em uma equipe formada pela pesquisadora, um professor e dois estudantes do Mestrado em Engenharia Elétrica da PUC Minas.

Os resultados mostraram que, ao considerar a terceira dimensão no estudo da área de vida do animais, foi possível descobrir novas informações até então imperceptíveis e que esta descoberta de informação é potencializada quando do uso de métodos mais apurados, como por exemplo, o uso do MPC3 ao invés do AM. Tais resultados motivaram o vislumbramento de trabalhos futuros, como a criação e a utilização de um método em três dimensões que seja baseado no Kernel, que dê informações mais precisas sobre a utilização do espaço baseada na densidade ou probabilidade de desta utilização.

Entre os trabalhos citados anteriormente, o trabalho de Howison; Bethel e Childs (2011) é o que mais se aproxima do trabalho aqui desenvolvido, tanto pela utilização de arquiteturas paralelas (uma das três arquiteturas utilizadas é igual à utilizada aqui), quanto pelo domínio onde estas arquiteturas foram utilizadas, a saber, a renderização volumétrica. A principal diferença apresentada é que Howison; Bethel e Childs (2011) constrói um comparativo entre as arquiteturas que utilizou, enquanto o objetivo aqui não foi comparar arquiteturas, mas sim explorar meios para se aproveitar ao máximo a capacidade que elas possuem. Já com relação ao trabalho de Bonde (2011), que é um trabalho relacionado ao domínio da aplicação, esta pesquisa estabelece uma forte relação de complementação, uma vez que aqui é desenvolvido e disponibilizado um novo método e uma ferramenta vislumbrada na referida pesquisa.

# 4 DESENVOLVIMENTO

Este capítulo discorre sobre os principais marcos do desenvolvimento do trabalho e os resultados alcançados.

## 4.1 Descrição do Estudo de Caso

O PPGZV realiza várias pesquisas que demandam recursos tecnológicos (hardware e software) para serem executadas. Uma destas pesquisas busca estudar o padrão de comportamento e o uso do *habitat* de alguns grupos de primatas que vivem na Reserva Particular do Patrimônio Nacional (RPPN) do Santuário do Caraça, que se localiza no município de Catas Altas – MG, a 120 km a leste de Belo Horizonte.

Um dos principais objetos da pesquisa ora mencionada refere-se ao uso da área de vida em três dimensões pelos animais (WORTON, 1989). O estudo inicial dedica-se aos primatas Guigós (KINZEY; COIMBRA-FILHO, 1981) (OLIVEIRA; COELHO; MELLO, 2003) através do projeto denominado “Projeto Guigó”, mas possui a perspectiva de extensão da tecnologia desenvolvida para estudar qualquer animal que utilize o espaço em três dimensões.

O estudo destes animais envolve a coleta de dados relacionados ao seu comportamento e localização. Uma pesquisadora acompanha os animais em seu *habitat* e coleta os dados necessários. Posteriormente, estes dados são processados para a criação de indicadores estatísticos e outras informações importantes para os pesquisadores.

São coletadas, para fins de análise quantitativa espacial, as coordenadas geográficas (latitude, longitude e altitude) em que os animais foram observados, além da altura que o animal encontra-se em relação ao solo. Para fins qualitativos, são coletadas informações sobre a identificação dos animais (nome, sexo, grupo, idade), o extrato em que se encontra (no chão ou na copa de uma árvore, por exemplo) e o comportamento apresentado no momento da observação.

O foco desta pesquisa abrange os dados espaciais coletados, no intuito de fornecer uma ferramenta que possibilite o processamento e a análise do uso do espaço de vida em três dimensões para os referidos animais. As três dimensões consideradas são a latitude, a longitude e altura em que o animal se encontra em relação ao solo, a altitude foi desconsiderada nesta primeira versão do modelo, tomando-se o terreno como plano.

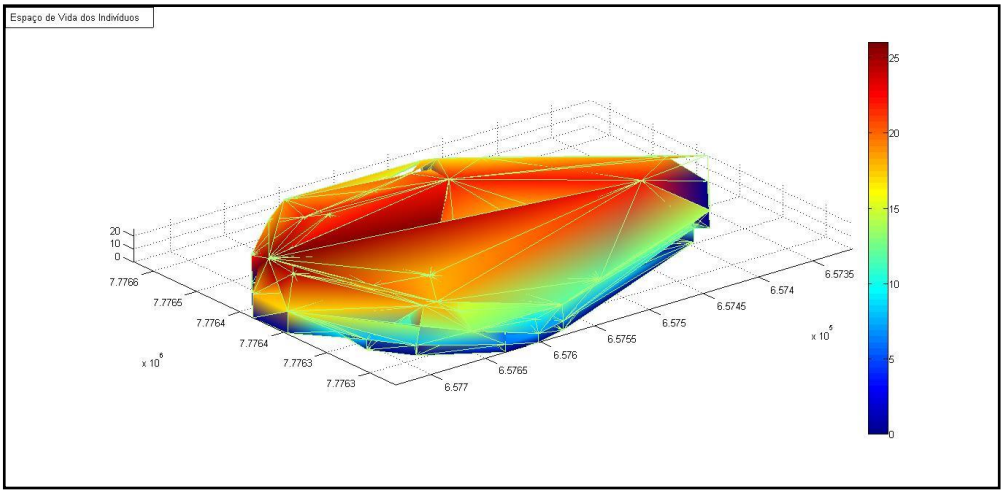
Para tal processamento, os pesquisadores do PPGZV precisam de ferramental computacional que muitas vezes não possuem e daí foi que surgiu a motivação para a parceria

firmada entre os programas de pós-graduação. A parceria entre os programas já existe e alguns resultados prévios já foram alcançados, destacando-se o trabalho de Bonde (2011), no qual a pesquisadora precisava estudar questões comportamentais dos animais, sendo uma delas a utilização do espaço tridimensional.

A equipe, formada pela pesquisadora e por alunos e professor do PPGEE, construiu um software *web* onde os dados coletados em campo são armazenados em uma base de dados via carregamento de arquivo. Para esses dados foram implementadas funcionalidades relacionadas às necessidades da pesquisa, entre elas, um método de cálculo e visualização do espaço tridimensional utilizado pelos animais.

O método de visualização implementado é bastante simples, baseado em mecanismos de análise bidimensional já existente, denominado Mínimo Polígono Convexo (MPC). A equipe estendeu o MPC para três dimensões, criando uma espécie de Mínimo Poliedro Convexo sobre o conjunto amostral. À partir dessa extensão, criaram-se mecanismos para o cálculo do volume do poliedro e para visualização da superfície dessa estrutura, exibindo o volume como um diamante manipulável, como exemplificado na Figura 6, uma imagem produzida pelo MatLab (MATLAB, 2011) ao executar um script gerado pelo software desenvolvido.

**Figura 6: Imagem criada no MatLab com script gerado pelo software desenvolvido**




**Fonte: Bonde (2011)**

No entanto, esse mecanismo precisou ser aprimorado, pois existe a carência de uma abordagem para análise do uso do espaço tridimensional mais refinada, que seja baseada no método Kernel (WORTON, 1989), um método mais preciso e confiável, já maduro, mas até então aplicado somente para duas dimensões na área de Zoologia.

O método Kernel considera que o espaço não é todo utilizado com a mesma intensidade, ao fazer a análise da utilização. Dessa forma, ele define meios para calcular a intensidade da utilização, baseados em amostras coletadas em campo. Portanto, é possível identificar regiões mais e menos utilizadas. As visualizações podem ser construídas com base em gráficos, atribuindo-se cores específicas para cada nível de intensidade de utilização.

No entanto, para este novo modelo de cálculo e visualização do uso da área de vida, a quantidade de dados para processar era grande e os algoritmos para geração dos valores de volume e geração das imagens representativas eram bastante complexos. Assim, o presente estudo de caso é uma instância do problema geral da pesquisa apresentado na seção 1.2, e pode ser apresentado com o questionamento de "como ter uma aplicação capaz de processar a massa de dados proveniente da pesquisa sobre os primatas e gerar, em tempo hábil e qualidade aceitável, resultados numéricos e resultados visuais que representem o volume ocupado pelos animais pesquisados?".

No tocante à busca da solução para o problema e como continuação da parceria entre os programas de pós-graduação, algumas etapas foram desenvolvidas para atender às novas demandas. Entre elas, a extensão do método Kernel para um método equivalente em três dimensões, juntamente com mecanismos para apresentação dos seus resultados. Os principais itens desenvolvidos foram:

- O modelo matemático para construção e cálculo de área de vida, baseado no Kernel em duas dimensões;
- O algoritmo referente ao modelo matemático definido;
- Uma aplicação computacional que:
    - implementou o referido cálculo de área de vida, fazendo-o através da utilização de computação paralela em GPU com CUDA;
    - implementou um mecanismo simplificado para apresentação visual desse volume, com a utilização de OpenGL[3].

[3] OpenGL (*Open Graphics Library*) é uma API (*Application Programming Interface*) aberta e multiplataforma de rotinas gráficas e de modelagem utilizada para o desenvolvimento de aplicações de Computação Gráfica (KHRONOS GROUP, 2013).

## 4.2 O modelo matemático para *Kernel* tridimensional

O método Kernel busca estimar o uso de todos os pontos de um determinado espaço com base em alguns pontos amostrados ou observados neste espaço.

De acordo com HU-BERLIM (2012), o método *Kernel* para estimação de distribuição de utilização do espaço de vida pode ser representado pelo modelo geral a seguir:

$$\hat{f}_{h(x)} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h1 \dots hd}K\left(\frac{(x1 - Xi1)}{h1}, \dots, \frac{(xd - Xid)}{hd}\right) \quad (1)$$

Nesse modelo, *K* é uma função de Kernel que será escolhida posteriormente, *h* é um vetor contendo a largura da banda a ser considerada para o cálculo da distribuição em cada dimensão, *d* é o número de dimensões consideradas e *n* é a quantidade de pontos amostrais que serão processados para a geração da estimativa.

A técnica chamada Kernel Multiplicativo é forma mais comum para a solução desse modelo geral. Ela consiste em desmembrar a função *K*, resolvendo-a para cada dimensão considerada. Multiplicando-se as funções desmembradas chega-se ao resultado final, representado pela seguinte equação geral:

$$\hat{f}_{h(x)} = \frac{1}{n}\sum_{i=1}^{n}\left\{\prod_{j=1}^{d}\frac{1}{hj}K\left(\frac{(xj - Xij)}{hj}\right)\right\} \quad (2)$$

Pelo foco desse trabalho, será considerado o cálculo da distribuição da utilização da área de vida em três dimensões e, dessa forma, sendo *d=3*, emprega-se a seguinte equação:

$$\hat{f}_{h(x)} = \hat{f}_{h1h2h3(x1,x2,x3)}$$
$$= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h1}\frac{1}{h2}\frac{1}{h3}K\left(\frac{(x1 - Xi1)}{h1}\right)K\left(\frac{(x2 - Xi2)}{h2}\right)K\left(\frac{(x3 - Xi3)}{h3}\right) \quad (3)$$

No entanto, tem-se ainda a função *K* que não foi determinada. Com base em Wand e Jones (1995) e em Fisher (2012), uma boa alternativa para resolver a função *K* é através de uma função de Kernel Epanechnikov, pela sua simplicidade e por apresentar-se mais eficiente no que se refere à estimativa da distribuição de densidades de ocupação pelas regiões do espaço considerado. Essa alternativa aplicada à equação anterior transforma-a na equação

seguinte, versão utilizada neste trabalho para o cálculo de distribuição da utilização do espaço em três dimensões:

$$
\begin{aligned}
&\hat{\hat{f}}_{h(x)} = \hat{\hat{f}}_{h1h2h3(x1,x2,x3)} \\
&= \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h1}\frac{1}{h2}\frac{1}{h3} \\
&* \frac{3}{4}\left(1 - \left(\frac{(x1 - Xi1)}{h1}\right)^2\right) I\left(\left|\frac{xi - Xi1}{h1}\right| \leq 1\right) \\
&* \frac{3}{4}\left(1 - \left(\frac{(x2 - Xi2)}{h2}\right)^2\right) I\left(\left|\frac{xi - Xi2}{h2}\right| \leq 1\right) \\
&* \frac{3}{4}\left(1 - \left(\frac{(x3 - Xi3)}{h3}\right)^2\right) I\left(\left|\frac{xi - Xi3}{h3}\right| \leq 1\right)
\end{aligned}
\tag{4}
$$

A função *I* é responsável por delimitar quais pontos amostrais contribuem para a estimativa da densidade de utilização em cada célula, fazendo isso com base na distância entre o ponto a ser estimado e ponto amostral corrente, comparando essa distância com o valor definido para o fator *h*, já apresentado anteriormente.

O fator *h* utilizado para cada dimensão deve ser definido pelo pesquisador no momento em que for submeter seus conjuntos de dados para o processamento. Para as execuções experimentais deste trabalho utilizou-se um valor constante para as dimensões consederadas.

O modelo matemático final para o cálculo da área de vida em três dimensões baseado no método Kernel pode, com base na equação anterior, ser apresentado na forma algorítmica através do pseudocódigo apresentado na seguinte descrição:

**Quadro 1: Algoritmo do Cálculo Kernel 3D sequencial**

```
Algoritmo: Cálculo Kernel 3D sequencial:
   1. Leia e armazene os pontos amostrais em P;
   2. Leia e armazene o Fator h em H;
   3. Identifique a largura de cada uma das dimensões dos
      pontos amostrais e armazene-as em DX, DY, DZ;
   4. Crie uma matriz tridimensional G com largura, altura
      e profundidade igual a DX, DY e DZ, respectivamente;
```

```
   5. Para cada célula da matriz G, calcule o valor da
      estimativa de utilização:
      5.1. Inicialize o valor de intensidade K com 0;
      5.2. Percorra o conjunto de pontos amostrais P
         calculando a contribuição de cada um deles para o
         valor de intensidade de utilização, conforme
         equação (4), acumulando-a em K;
      5.3. Armazene o valor de K na posição corrente da
         matriz G.
   6. Exiba o resultado do cálculo da área de vida;
Fim do Algoritmo.
```

**Fonte: Elaborado pelo autor**

O cálculo, como pode ser observado no pseudocódigo acima, é exaustivo, pesado e possui a característica de ser altamente paralelizável. O cálculo da intensidade de cada célula da matriz é independente das demais células da matriz, dependendo apenas do conjunto de pontos amostrais, que não recebe nenhuma alteração durante o processamento e pode ser compartilhado entre os elementos de processamento sem nenhum risco para a aplicação. A versão paralela do algoritmo é apresentada na seção seguinte.

## 4.3 O algoritmo paralelo

Como o objetivo desta pesquisa aborda a exploração da aplicação da computação paralela na Visualização Científica, fez-se necessário criar uma versão paralela do algoritmo apresentado na seção anterior. Tal algoritmo foi desenvolvido considerando-se a tecnologia na qual seria implementado, que é a plataforma de programação CUDA. O pseudocódigo pode ser observado a seguir:

**Quadro 1: Algoritmo do Cálculo Kernel 3D paralelo**

```
Algoritmo: Cálculo Kernel 3D paralelo:
   1. Leia e armazene os pontos amostrais em P;
   2. Leia e armazene o fator h em H;
   3. Identifique a largura de cada uma das dimensões dos
```

pontos amostrais e armazene-as em DX, DY, DZ;
4. Crie uma matriz tridimensional G com largura, altura e profundidade igual a DX, DY e DZ, respectivamente;
5. Defina a quantidade de *threads* T;
6. Defina a quantidade de células Q a serem processadas por cada *thread*;
7. Transfira os pontos amostrais P para a memória compartilhada da placa gráfica;
8. Aloque um espaço M equivalente ao tamanho da matriz G na memória da placa gráfica;
9. Invoque a execução *multithread* na GPU da placa gráfica:
   9.1. Para cada *thread* em execução na placa gráfica:
      9.1.1. Recupere o identificador[4] I do *thread*;
      9.1.2. Defina, com base em I e em Q, quais células serão processadas pela *thread*;
      9.1.3. Para cada célula do escopo do *thread*, calcule o valor da estimativa de utilização:
         9.1.3.1. Inicialize o valor de intensidade K com zero;
         9.1.3.2. Percorra o conjunto de pontos amostrais P calculando a contribuição de cada um deles para o valor de intensidade de utilização, conforme equação (4), acumulando-a em K;
         9.1.3.3. Armazene o valor de K na posição corrente da memória M.
10. Copie o conteúdo da memória M para a matriz G.
11. Exiba o resultado do cálculo da área de vida;

Fim do Algoritmo.

**Fonte: Elaborada pelo autor**

[4] Com CUDA, cada *thread* em execução em uma GPU recebe um identificador único que pode ser utilizado para algum propósito específico da aplicação.

A versão paralela do algoritmo desenvolvido para a plataforma CUDA apresenta um nível de complexidade significativamente maior que a versão sequencial. Isto porque envolve aspectos como alocação de memória na placa gráfica, transferência de dados da memória RAM do computador para a memória da placa gráfica e vice versa, invocação da execução paralela, entre outros.

## 4.4 A aplicação construída

As versões do software foram construídas utilizando-se a plataforma de desenvolvimento Microsoft Visual Studio 2010, ferramenta bastante completa e produtiva, disponível nos laboratórios de informática da PUC Minas. Foi utilizada a linguagem CUDA C, uma linguagem baseada em C, mas com algumas restrições necessárias para torná-la executável em GPUs da NVidia.

Além da linguagem CUDA C, também foram necessários o NVIDIA CUDA *Toolkit* e o NVIDIA GPU *Computing* SDK. O NVIDIA CUDA *Toolkit* provê o ambiente necessário para os desenvolvedores C e C++ criarem aplicações para GPUs. Ele inclui compiladores, bibliotecas, ferramentas para *debugging* e otimização etc. Já o NVIDIA GPU *Computing* SDK traz uma grande quantidade de exemplos e esqueletos de projetos que ajudam os desenvolvedores a escreverem aplicações com CUDA C/C++.

A aplicação foi construída para o ambiente Windows, utilizando-se o padrão "*Console Application*". Significa que suas saídas são feitas através do *prompt* de comando do Windows, exceto a imagem representativa da distribuição da área de vida tridimensional, que é apresentada em uma janela criada através de chamadas às funções da biblioteca gráfica OpenGL (KHRONOS GROUP, 2013) e outras bibliotecas que trabalham em conjunto com esta (a saber, as bibliotecas GLU e GLUT, que são bibliotecas gratuitas e de código aberto, amplamente utilizadas para visualização).

Foram construídas, considerando os experimentos a serem feitos, quatro versões da aplicação. Uma primeira adaptada para executar o cálculo da distribuição da área de vida de forma sequencial em CPU, uma segunda adaptada para executar o cálculo de forma paralela em CPU, uma terceira adaptada para execução do cálculo em paralelo em GPU, compartilhada com sistema operacional da máquina, e uma quarta adaptada para a execução do cálculo em GPU, dedicada somente ao cálculo. As versões de teste serão melhor detalhadas no próximo capítulo.

Como ilustração, são apresentados nas Figuras 7 e 8, a seguir, os trechos de código referentes ao cálculo do Kernel para as versões sequencial em CPU e paralela em GPU, respectivamente. É importante destacar que o cálculo executado é o mesmo nas duas versões do código, porém, no primeiro, a seção referente ao cálculo acontece dentro de laços de repetição que, sequencialmente, executam os cálculos para cada uma das células da matriz de saída. Por outro lado, no segundo código, não existem laços de repetição para varredura da matriz de saída, cada *thread* executa em paralelo sobre as saídas que lhe foram atribuídas através da distribuição da quantidade de blocos e *threads*.

**Figura 7: Codificação em Linguagem C da versão sequencial do cálculo**

```
void kernelSequencial(){
    float kvalue, dist, _1sobreH, _1sobreN;
    int i, j, k, p, aux;
    Voxel v;
    _1sobreH = (float) 1 / fatorh;
    _1sobreN = (float) 1 / npontos;
    for(i = 0; i < tamx; i ++){
        for(j = 0; j < tamy; j ++){
            for(k = 0; k < tamz; k ++){
                kvalue = 0.0f;
                dist = 0.0f;
                for (p = 0; p < npontos; p++){
                    aux = p * dimensao;
                    if(fabs((i - pontos->valores[aux])) <= fatorh){
                        dist = 0.75 * (1 - pow((i - pontos->valores[aux]) / fatorh, 2)) * (fabs((i - pontos->valores[aux])) <= fatorh);
                        dist *= (0.75 * (1 - pow((j - pontos->valores[aux+1]) / fatorh, 2)) * (fabs((j - pontos->valores[aux+1])) <= fatorh));
                        dist *= (0.75 * (1 - pow((k - pontos->valores[aux+2]) / fatorh, 2)) * (fabs((k - pontos->valores[aux+2])) <= fatorh));
                    } else
                        dist = 0;
                    kvalue += dist;
                }
                kvalue = (kvalue * _1sobreN * _1sobreH * _1sobreH * _1sobreH);
                v.densidade = kvalue;
                gride->matriz[i][j][k] = v;
    }   }   }
}
```

**Fonte: Elaborada pelo autor**

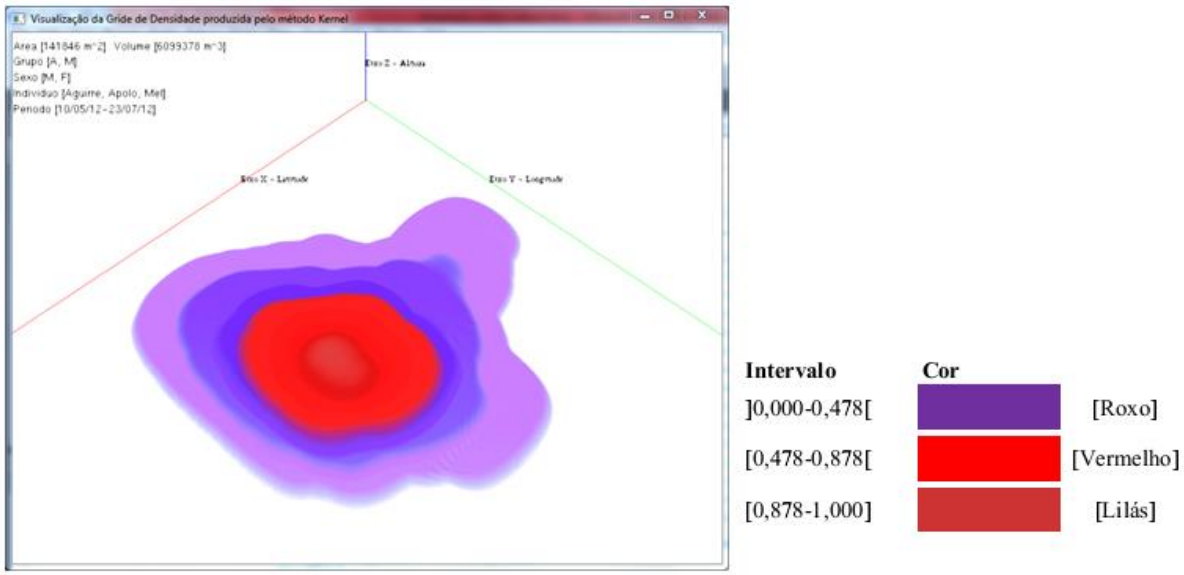
**Figura 8: Codificação em Linguagem CUDA C da versão sequencial do cálculo**

```
__global__ void kernelParaleloGPU(float* gridekernel, float* points, int tamx, int tamy, int tamz,
                                  int npontos, float fatorh){
    int aux;
    float kvalue, dist;
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    int j = blockIdx.y * blockDim.y + threadIdx.y;
    if(i < tamx && j < tamy){
        for(int k = 0; k < tamz; k++){
            kvalue = 0.0f;
            dist = 0.0f;
            for (int p = 0; p < npontos; p++){
                aux = p * 4;
                if(fabs((i - points[aux])) <= fatorh){
                    dist = 0.75f * (1.0f - pow((i - points[aux]) / fatorh, 2)) * (fabs((i - points[aux])) <= fatorh);
                    dist *= (0.75f * (1.0f - pow((j - points[aux+1]) / fatorh, 2)) * (fabs((j - points[aux+1])) <= fatorh));
                    dist *= (0.75f * (1.0f - pow((k - points[aux+2]) / fatorh, 2)) * (fabs((k - points[aux+2])) <= fatorh));
                }else
                    dist = 0;
                kvalue += dist;
            }
            kvalue = (kvalue * (1.0f / npontos) * (1.0f / fatorh) * (1.0f / fatorh) * (1.0f / fatorh));
            gridekernel[(k + (j*tamz) + (i*tamz*tamy))] = kvalue;
    }   }
}
```

**Fonte: Elaborada pelo autor**

Todas as versões do software, em suas respectivas plataformas, geraram o resultado do processamento como esperado, ou seja, uma imagem representativa da distribuição da utilização do espaço tridimensional, como exemplificado na Figura 9, bem como as informações sobre o volume desse espaço. Elas apresentaram, porém, tempos de respostas significativamente diferentes para geração desses resultados.

**Figura 9: Visão superior da distribuição da utilização para o conjunto de dados 2**




**Fonte: Elaborada pelo autor**

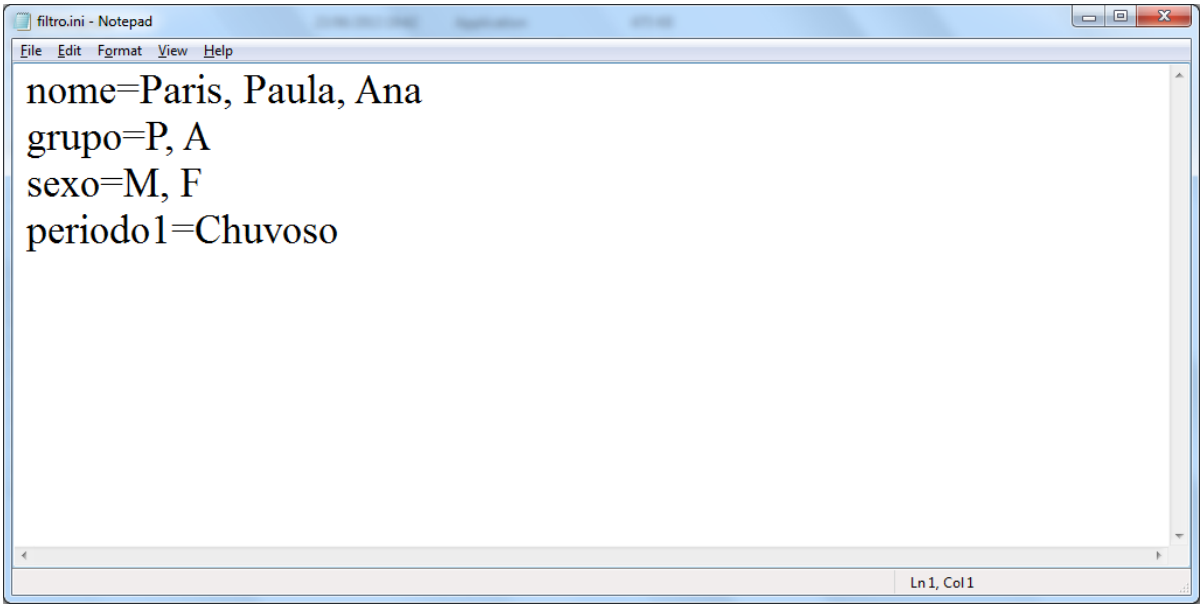
Na Figura 9, as cores representam a intensidade de utilização em cada região do volume e foram definidas para atender uma solicitação dos pesquisadores do PPGZV. Para cada célula da matriz tridimensional, desenhou-se um cubo unitário (um pixel de aresta) no espaço de visualização. A cor e a transparência de cada cubo foram definidas com base no valor estimado da distribuição de utilização da célula corrente, ou seja, ambos os atributos da imagem, cor e transparência, foram utilizados para comunicar de forma mais eficiente a mesma informação (densidade de ocupação). Para a cor foram definidos três possíveis valores, e para a transparência fez-se uma correspondência direta com o valor estimado da intensidade de ocupação da célula.

As possíveis cores e seus respectivos códigos RGB em OpenGL foram roxo [0.3, 0.0, 1.0], vermelho [1.0, 0.0, 0.0] e lilás [0.8, 0.2, 0.2]. Na Figura 4, nota-se a existência de diferentes tons de roxo, o que é resultado somente da transparência atribuída a cada elemento, uma vez que a mesma cor é mantida para cada um deles. O vermelho é mais constante e não

sofre muito o efeito da variação de transparência, o que acontece também com a região em lilás, ao cento da imagem.

Para a utilização da ferramenta, o pesquisador deve navegar até o diretório onde o arquivo executável se encontra, disponibilizar nesse diretório um arquivo contendo os dados a serem processados e configurar os parâmetros de sua análise através de um arquivo de configuração presente no mesmo diretório, exemplificado na Figura 10.

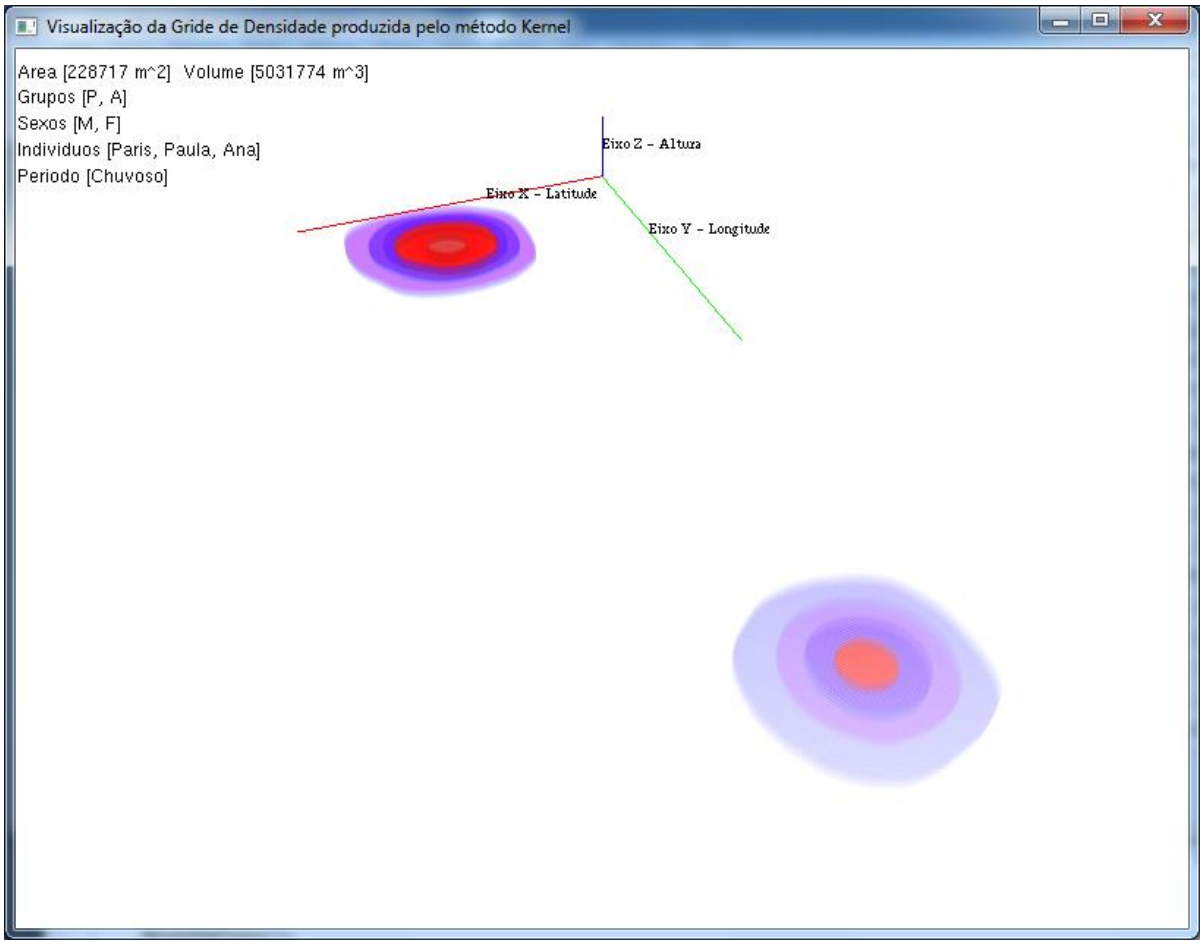
**Figura 10: Exemplo de filtro para a execução do software de cálculo**




**Fonte: Elaborada pelo autor**

Uma vez definidos os filtros, basta executar normalmente o programa. Este apresenta saídas textuais de controle via o *prompt* de comando do Windows e, ao final da execução, cria uma janela OpenGL com a imagem representativa do espaço de vida utilizado para o conjunto de pontos processados. Com os dados reais da pesquisa de Bonde (2011) e o filtro definido acima, o software gera como resultado a imagem ilustrada na Figura 11 utilizando, assim como os demais exemplos na sequência desta seção, o mesmo esquema de cores especificado com a Figura 9 apresentada anteriormente.

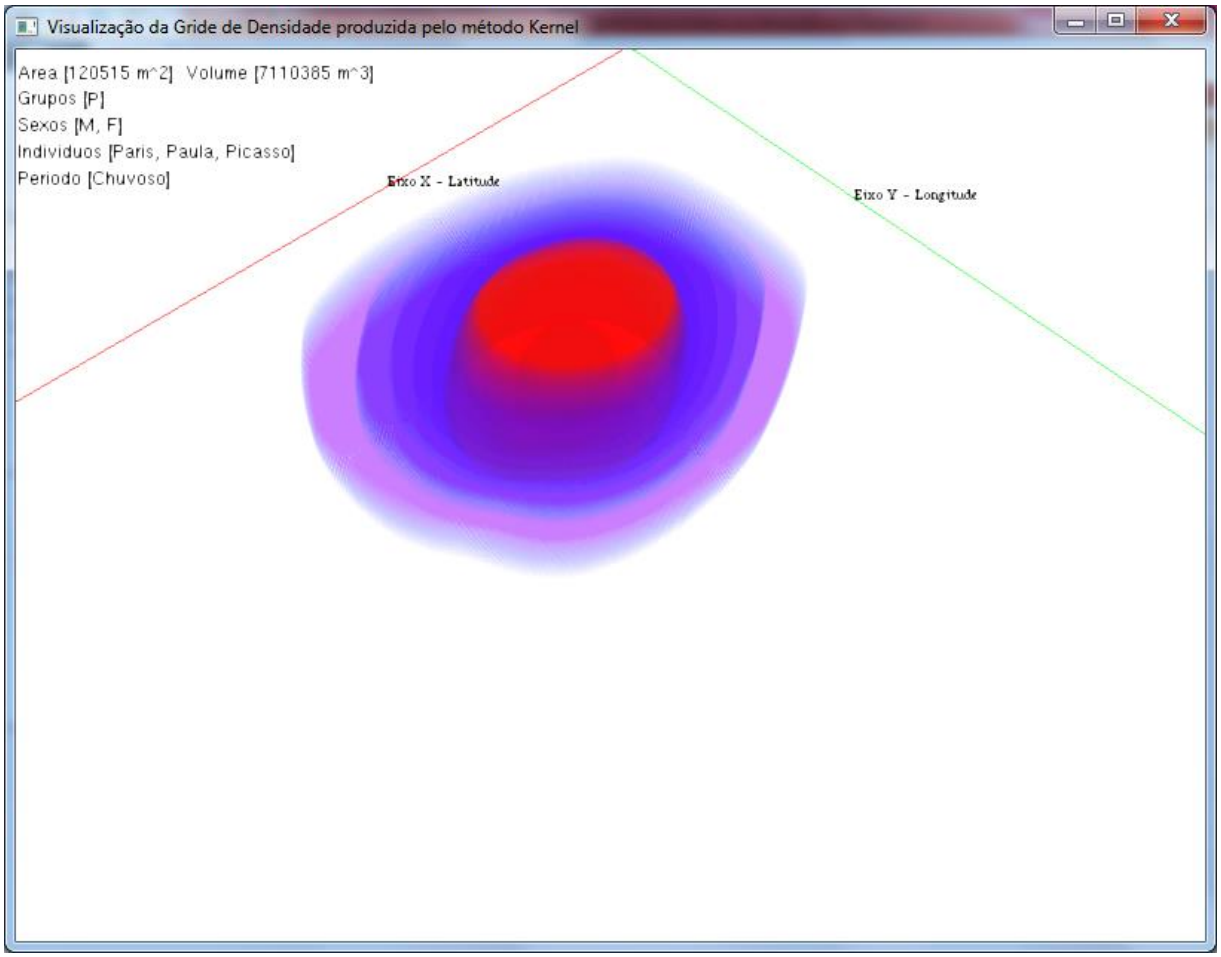
**Figura 11: Representação do espaço de vida utilizado pelos animais Paris, Paula e Ana no período chuvoso**




**Fonte: Elaborada pelo autor**

Percebe-se, pelo filtro descrito no campo textual e pela observação da imagem resultante, que o espaço de vida calculado e apresentado refere-se a indivíduos pertencentes a grupos diferentes, a saber , Grupo P e Grupo A. Admitindo-se que o pesquisador se interesse em avaliar cada um dos grupos separadamente e então analisar as diferenças, ele deve alterar o arquivo de filtro e informar os parâmetros de acordo com sua necessidade, por exemplo, inserindo uma configuração para que se considerem somente animais do grupo P. Os resultados provenientes dessa configuração geram o resultado apresentado na Figura 12.

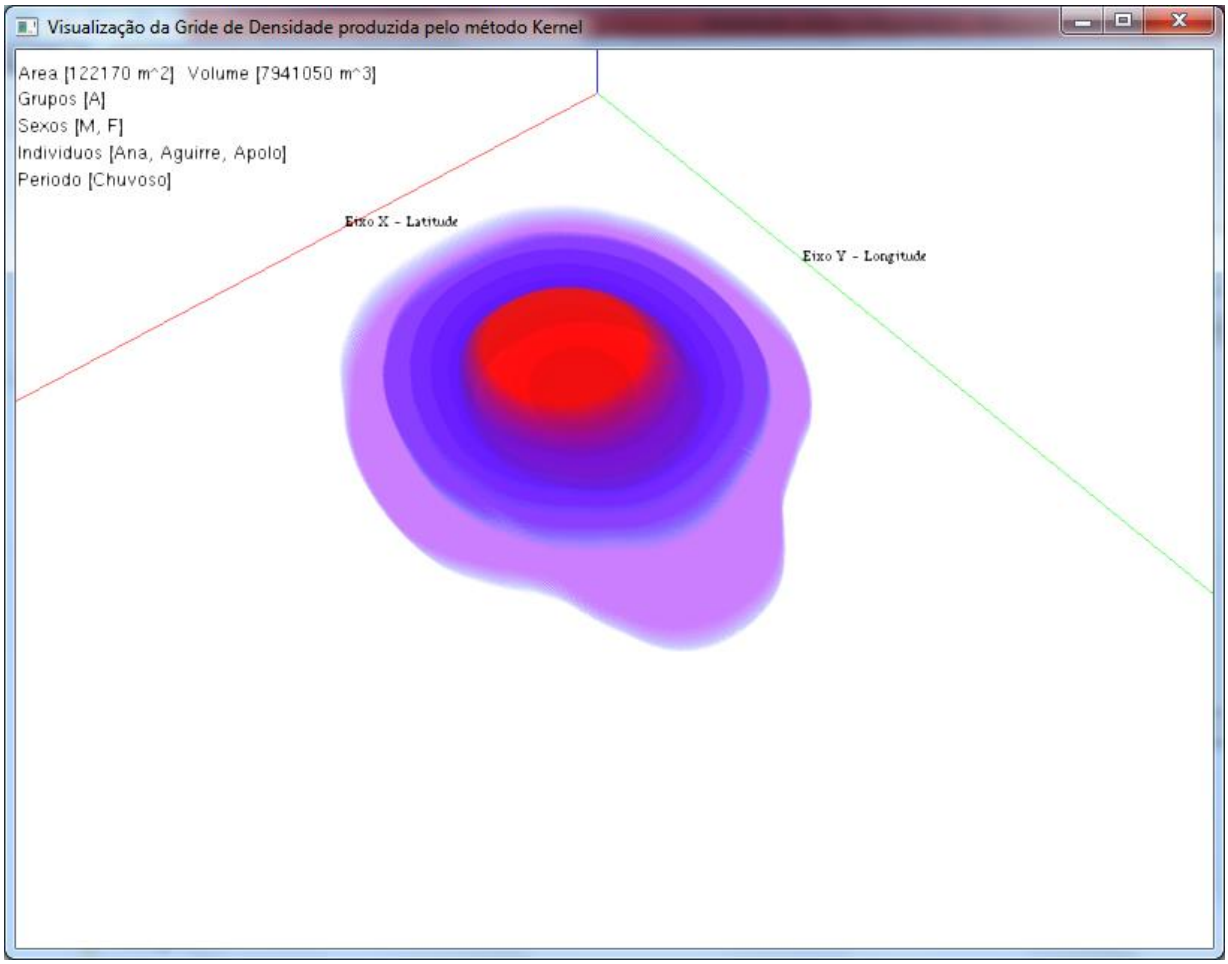
**Figura 12: Representação do espaço de vida utilizado pelos animais Paris, Paula e Picasso no período chuvoso**




**Fonte: Elaborada pelo autor**

Por outro lado, ao configurar o arquivo de parâmetros para que se considere somente o Grupo A, os resultados são os apresentados na Figura 13. Note que ao variar a amostra, os valores numéricos para área e volume sofrem alterações, estas alterações são geradas devido à natureza dos pontos amostrais, a forma como estão distribuídos, por exemplo, e também pelo tamanho da amostra considerada.

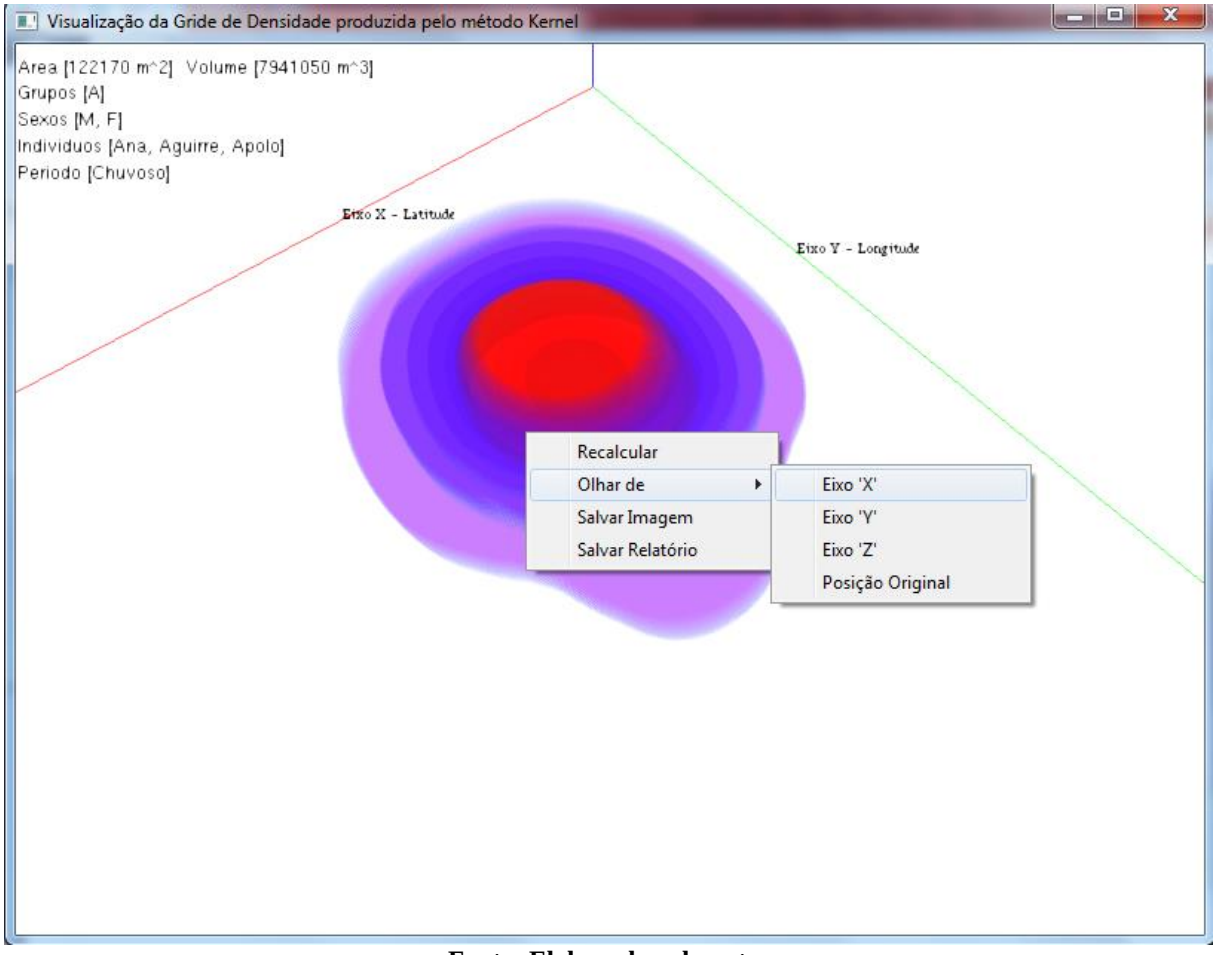
**Figura 13: Representação do espaço de vida utilizado pelos animais Ana, Aguirre e Apolo no período chuvoso**




**Fonte: Elaborada pelo autor**

A compreensão de estruturas volumétricas, muitas vezes, é melhor compreendida quando se permite alguma interação com o objeto visualizado. Nesse sentido, algumas formas de interação simplificadas foram desenvolvidas para facilitar esta compreensão. O pesquisador pode, por exemplo, mudar a posição de onde observa a estrutura volumétrica, o que gera uma alteração na imagem final projetada. Um *popup menu* construído em OpenGL permite esta alteração do ponto de vista do observador para alguns pontos predefinidos do espaço de visualização, o que pode ser ilustrado com as Figuras 14 e 15 que seguem.

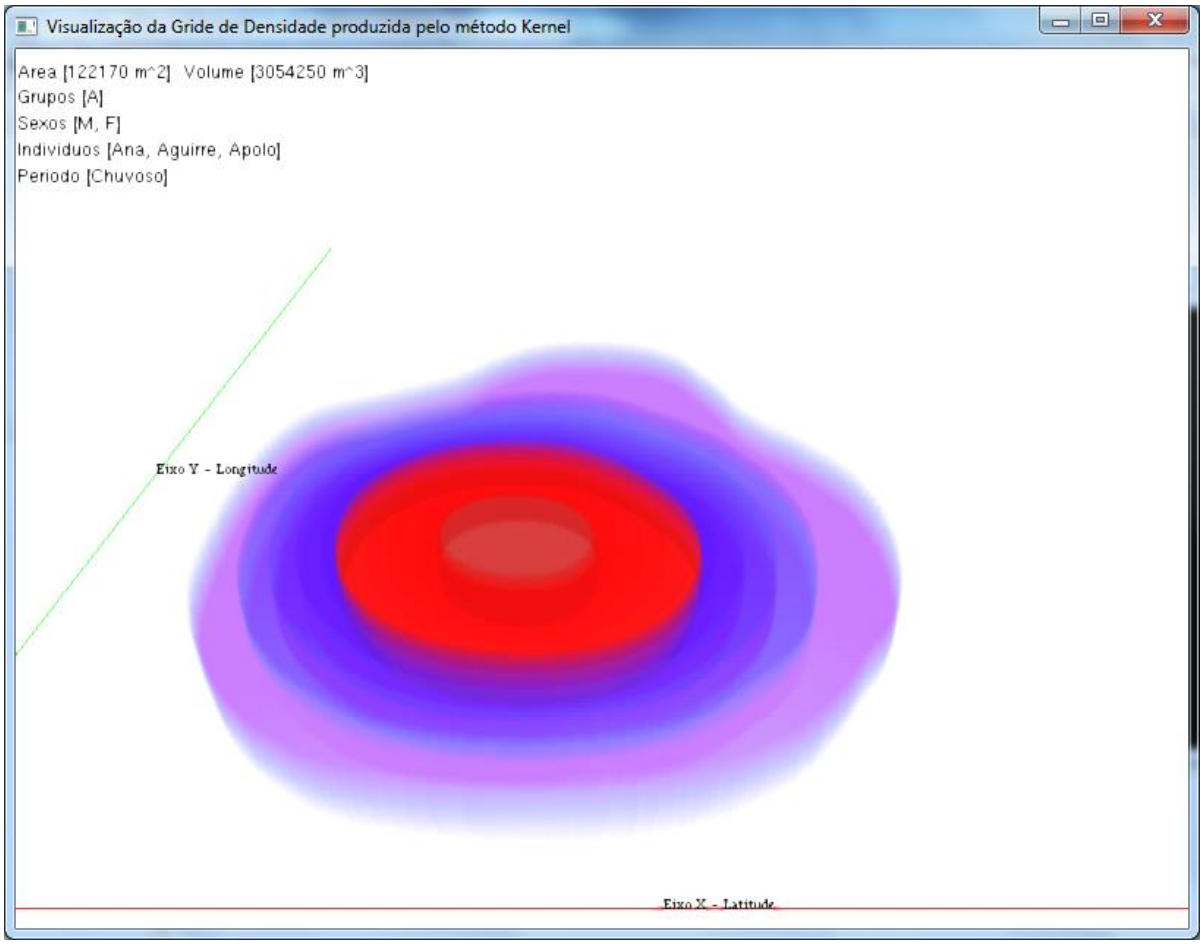
**Figura 14: Visão do *popup menu* de seleção da posição de visualização e outras opções**




**Fonte: Elaborada pelo autor**

O *popup menu* é um menu que não fica acoplado estaticamente em uma determinada posição da página. Pelo contrário, ele é acionado com clique do botão direito do *mouse*, surgindo na página no local onde o evento do clique aconteceu.

**Figura 15: Representação do espaço de vida utilizado pelo animais Ana, Aguirre e Apolo no período chuvoso, observados da direção do eixo X**




**Fonte: Elaborada pelo autor**

Percebe-se, pela Figura 15, que o ponto de vista utilizado para gerar a imagem influencia na visualização final gerada.

# 5 RESULTADOS EXPERIMENTAIS

Este capítulo apresenta e discute aos resultados obtidos com a experimentação da aplicação nas plataformas de computação selecionadas.

## 5.1 Ambiente experimental

Foram utilizadas quatro plataformas distintas para execução experimental do software construído. Estas plataformas foram necessárias para a coleta e análise das informações de desempenho da aplicação. A primeira plataforma possui como componente fundamental o Intel Core i3, uma CPU *multicore* de dois núcleos físicos e quatro núcleos lógicos, o que é possível com o uso da tecnologia *Hyperthreading*[5] da Intel, companhia que fabrica o processador. Nessa plataforma foram executadas a versão sequencial e a versão paralela para CPU da aplicação (tais versões serão abordadas mais à frente, na seção 5.3, que faz um apanhado geral de todas as versões construídas). A segunda plataforma tem como principal componente a GTX 260, uma GPU com capacidades medianas se comparadas com as GPUs mais avançadas do mercado atualmente. A terceira plataforma também é composta por uma GPU, a GTX 560, porém trata-se de uma GPU com características mais avançadas e com maiores capacidades, também como forma de confrontar e reforçar a análise de desempenho da aplicação. Por fim, a quarta plataforma possui duas GPUs como componentes básicos, sendo a primeira delas uma GPU de menor capacidade, que ficou responsável pelas tarefas do sistema operacional Windows 7 *Ultimate* 64 *bits*, e a segunda uma GPU mais avançada, a mesma utilizada na terceira plataforma testada, que ficou, portanto, responsável exclusivamente pela execução dos cálculos da aplicação, sem interrupções para o processamento de tarefas do sistema operacional, como movimento de janelas, entre outras tarefas de processamento gráfico geral do computador. Estas plataformas foram selecionadas considerando-se as CPUs ou GPUs nelas contidas, as necessidades arquiteturais relacionadas à experimentação e restrições financeiras. Um resumo das principais características dessas plataformas pode ser visualizado nas Tabelas 1 e 2 a seguir.

[5] Hyper-Threading é uma tecnologia da Intel que simula, em um processador físico, dois processadores lógicos (INTEL, 2013).

**Tabela 1: Resumo das características das GPUs utilizadas para o cálculo**

| Características | GTX 260 | GTX 560 |
|---|---|---|
| Capacidade da memória global (MB) | 892 | 1024 |
| Tipo da memória global | GDDR3 | GDDR5 |
| Frequência da memória global (MHz) | 1100 | 2002-2200 |
| Largura do barramento de acesso à memória global (bits) | 448 | 256 |
| Taxa máxima de transferência (GB/s) | 123,2 | 128 |
| Número de núcleos | 216 | 336 |
| Frequência dos núcleos (MHz) | 625 | 1620-1900 |
| Taxa máxima de execução de operações de ponto flutuante (GFLOPS) | 270 | 1262 |
| CUDA Capacidade computacional (versão) | 1.3 | 2.1 |
| Número máximo de *threads* por bloco | 512 | 1024 |
| Memória compartilhada por bloco (KB) | 16 | 48 |
| Memória constante (KB) | 64 | 64 |
| Número de registradores por bloco | 16384 | 32768 |
| Ponto flutuante de precisão dupla | Sim | Sim |

**Fonte: Elaborada pelo autor**

**Tabela 2: Resumo das características da CPU utilizada**

| Características | Intel Core i3 |
|---|---|
| Número de cores | 2 |
| Número de *threads* | 4 |
| Frequência de clock (GHz) | 2.3 |
| Memória *cache* disponível (MB) | 3 |
| Conjunto de instruções | 64-bit |
| Máxima energia térmica de projeto - TDP (W) | 35 |
| Tamanho máximo de memória (GB) | 16 |
| Memória RAM disponível (GB) | 4 |
| Tipos de memória RAM | DDR3-1066/1333 |
| Número de canais de memória | 2 |
| Largura de banda máxima de memória (GB/s) | 21,3 |

**Fonte: Elaborada pelo autor**

## 5.2 Cargas de trabalho

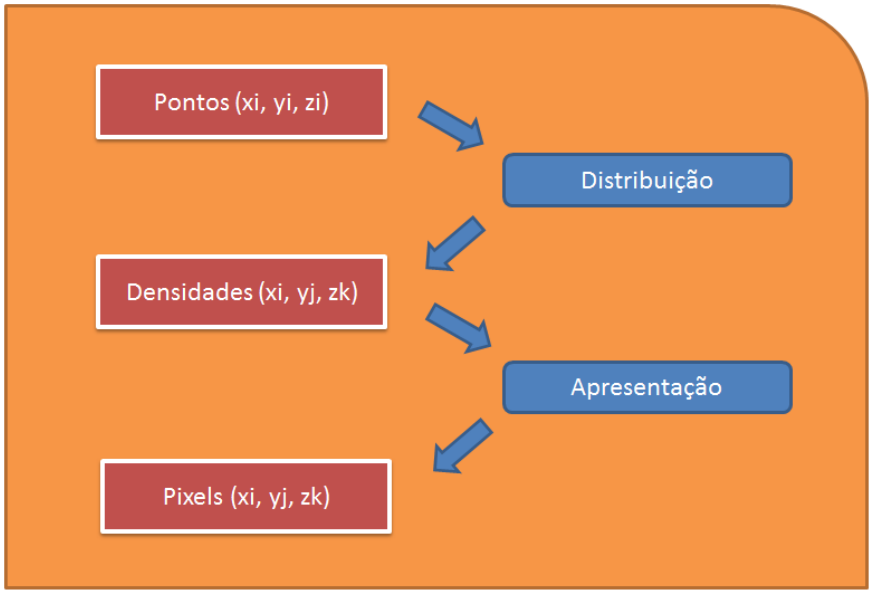
Uma carga de trabalho, em computação, refere-se ao conjunto de trabalhos que são enviados para uma unidade ou um conjunto de unidades de processamento. Na carga de

trabalho estão contidos os dados a serem processados e os algoritmos a serem executados sobre os dados.

As cargas de trabalho utilizadas nesta pesquisa foram construídas tomando-se conjuntos de pontos tridimensionais, que representam as posições (latitude, longitude e altura) onde os primatas foram observados em seu *habitat* natural, e os algoritmos de processamento desses dados, sendo o principal deles o cálculo de Kernel para estimação da distribuição da utilização da área de vida.

A aplicação construída pode ser dividida em três blocos básicos: o carregamento dos dados, o cálculo de *kernel* e a apresentação do resultado do cálculo em forma de imagens e valores quantitativos. A Figura 16 mostra o fluxo dos dados e as operações às quais esses dados são submetidos para se chegar ao resultado da visualização.

**Figura 16: Transformações sobre dados para geração das visualizações**




**Fonte: Elaborada pelo autor**

Pela imagem da Figura 16 é possível notar o fluxo dos dados na aplicação construída. Os dados amostrais, na forma de uma lista de pontos tridimensionais, são importados pela aplicação. A partir desses dados, faz-se o cálculo da utilização do volume, o que gera uma matriz tridimensional preenchida com valores de intensidade de utilização (densidade) para cada uma de suas células. Dessa matriz, afere-se o valor de volume com um somatório simples das células ocupadas e gera-se a imagem, projetando-se a matriz de densidades em um plano.

A matriz tridimensional é divida em células pelo fato de que não se chegará á uma modelagem contínua para a estimação e sim uma representação discreta, desta forma cada ponto tridimensional representa uma célula cúbica unitária, na qual o valor estimado será constante.

Como experimentos, foram utilizados três conjuntos de dados. Esses dados são oriundos da pesquisa de Bonde (2011), já utilizados em outras etapas da parceria dos programas de pós-graduação em questão. Tais dados foram resultados de observações feitas durante o período de um ano, período no qual a pesquisadora acompanhou alguns grupos de primatas no local onde desenvolveu sua pesquisa. Do conjunto total de dados retiraram-se três amostras para experimentação, sendo uma composta por todas as observações feitas e duas por subconjuntos aleatórios do conjunto original. O resumo dos conjuntos de dados pode ser observado na Tabela 3.

**Tabela 3: Conjuntos de dados utilizados**

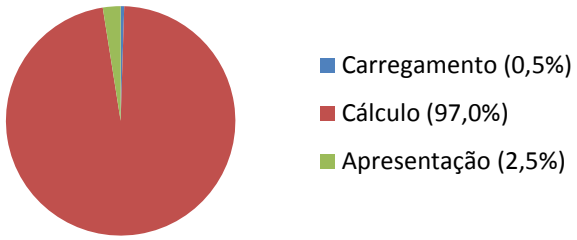
| **Conjunto de dados** | **Atributos** | | |
|---|---|---|---|
| | Número de pontos | Tamanho da matriz gerada | Número de Operações |
| **Conjunto 1** | 67 | 433 x 468 x 23 | 312.274.404 |
| **Conjunto 2** | 805 | 433 x 522 x 23 | 4.184.871.390 |
| **Conjunto 3** | 2.607 | 1078 x 2245 x 26 | 164.039.896.020 |

**Fonte: Elaborada pelo autor**

A tabela apresenta os três conjuntos de dados caracterizados por três atributos específicos. O atributo “número de pontos” refere-se à quantidade de pontos amostrais contidos no conjunto. O atributo “tamanho da matriz gerada” refere-se ao tamanho da matriz tridimensional necessária para abranger todos os pontos da amostra (por analogia, esse atributo refere-se às dimensões de uma caixa necessária para cobrir todos os pontos amostrais. No ambiente natural, a medida correspondente está em metros, já no ambiente virtual, é transformada para *pixels*). Por fim, o atributo “número de operações” refere-se à quantidade de cálculos de contribuição que serão feitas para que a matriz tridimensional seja totalmente preenchida com valores estimados de densidade de utilização, esse número é obtido multiplicando-se a quantidade de pontos pela quantidade de células que compõem a matriz gerada. Cada cálculo de contribuição refere-se a uma iteração do somatório da equação final apresentada na seção 4.2 deste documento.

Das partes que compõem o sistema, ao considerar-se uma versão sequencial da aplicação e uma amostra de tamanho médio em relação às amostras utilizadas, tem-se o tempo de execução distribuído conforme o Gráfico 1. Pela distribuição dos tempos de execução apresentados no Gráfico 1, nota-se que o principal gargalo da aplicação encontra-se no cálculo da estimativa de utilização através do método Kernel. Esse é, portanto, a parte da aplicação com maior margem para melhorias de desempenho e é justamente nela que este trabalho empenhou esforços.

**Gráfico 1: Distribuição média dos tempos de execução do programa sequencial**




**Fonte: Elaborada pelo autor**

Como o foco deste trabalho é a paralelização do cálculo de Kernel, consideram-se sequenciais os demais componentes da aplicação. Feita essa consideração, pode-se aplicar a Lei de Amdahl (AMDAHL, 1967), que relaciona o *speedup* total da aplicação com o *speedup* obtido na fração paralelizada e a fração do tempo de execução sequencial da parte a ser paralelizada, conforme a equação abaixo, adaptada do referido autor.

$$S_{total} = \frac{1}{\left(1 - f_{paralelo}\right) + \frac{f_{paralelo}}{S}} \quad (5)$$

A lei de Amdahl estabelece dois princípios básicos sobre o ganho de desempenho. O primeiro define que quando a fração paralela de uma aplicação é pequena, o *speedup* total que pode ser alcançado também o será. Já o segundo estabelece que quando o *speedup* alcançado com a fração paralelizada tende ao infinito, o *speedup* total será limitado pela parte não paralelizada do programa. Dessa forma, o *speedup* máximo que pode ser alcançado neste trabalho é bastante significativo, uma vez que a parte paralelizável da aplicação é consideravelmente maior que as partes intrinsicamente sequenciais.

Vale ressaltar que a proporção dos tempos de execução varia consideravelmente em relação à quantidade de pontos da amostra processada e do tamanho da grade necessária para cobri-los. Isso deve-se ao fato de o tempo de cálculo variar com uma maior sensibilidade do que o tempo de execução do carregamento e da apresentação. Como os conjuntos de dados a serem processados são, geralmente, grandes, a tendência é de ganhos de desempenho ainda melhores que aqueles que se acredita alcançar com base na análise do Gráfico 1.

## 5.3 Versões do programa

Cada conjunto de dados foi submetido a cinco execuções distintas, sobre as quatro plataformas de computação selecionadas, com objetivo de confrontar os resultados com aqueles obtidos na arquitetura paralela alvo desta pesquisa. As referidas plataformas foram apresentadas na seção 5.1 deste documento.

Para que o software pudesse ser executado em cada uma das plataformas selecionadas, fez-se necessária a construção de quatro versões distintas do mesmo. A primeira versão é aquela que faz o cálculo sequencial em CPU, a segunda versão foi desenvolvida para executar o cálculo em paralelo também em CPU, a terceira versão foi desenvolvida para execução do cálculo em paralelo utilizando GPU, considerando o compartilhamento desse recurso com o sistema operacional e, por fim, a quarta versão foi desenvolvida para execução do cálculo em paralelo utilizando GPU, considerando o recurso exclusivo para execução dos cálculos, sem interferências advinda das demais tarefas do sistema.

A versão sequencial foi implementada para servir como objeto de comparação para as demais versões. Nessa versão, o cálculo de Kernel é simplesmente uma tradução para linguagem C do processamento definido na equação (4) e no pseudocódigo da seção 4.2.

A versão paralela para CPU foi desenvolvida com base na versão sequencial, utilizando-se a API OpenMP[6]. Esta API provê uma forma simplificada de desenvolvimento *multithreading* através de diretivas de compilação, biblioteca de rotinas de execução e variáveis de ambiente. Seu objetivo fundamental é permitir a padronização das aplicações, o desenvolvimento mais enxuto, a facilidade de uso e a portabilidade. As alterações sobre o programa sequencial, para torná-lo paralelo, são simplistas e não caracterizam um novo

[6] OpenMP é uma API padronizada para programação paralela em Fortran, C, e C++ que utiliza arquitetura de memória compartilhada (BARNEY, 2013).

algoritmo, pois é necessário apenas inserir diretivas OpenMP nas seções de código que se dejeva executar em paralelo.

A terceira e a quarta versão foram desenvolvidas com base no pseudocódigo apresentado na seção 4.3, utilizando-se a tecnologia CUDA. O cálculo de utilização do espaço de vida pelo Método Kernel foi implementado na forma da função *kernel* do programa em CUDA C. Foram verificados alguns arranjos de blocos e *threads* e os resultados apresentados na próxima seção referem-se aos arranjos mais vantajosos, no que se refere ao tempo de execução.

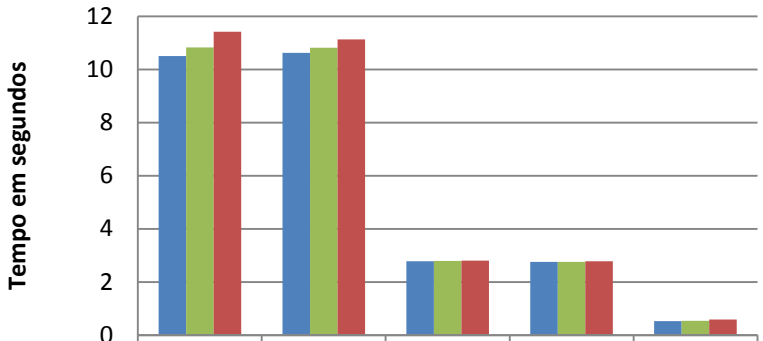
A diferença básica entre as duas últimas versões é que a terceira considera a GPU como recurso compartilhado com sistema operacional e a quarta considera tal recurso como de uso exclusivo. A primeira situação cria a exigência de que as porções de trabalho enviados para execução sejam menores. Isto porque o sistema operacional (neste caso, o Windows) estabelece um tempo máximo de ocupação da GPU e interrompe a aplicação caso ela estoure tal limite. Resolver o problema dessa maneira acaba desencadeando uma quantidade maior de chamadas de execução e comunicação de dados. Por outro lado, a segunda situação presume que a GPU é exclusiva para o uso da aplicação, então todo o trabalho pode ser enviado ao componente de uma única vez e não há limite de ocupação do seu recurso.

## 5.4 Experimentos realizados

Os experimentos foram divididos em cinco diferentes grupos. Inicialmente fez-se a execução da versão sequencial para CPU na plataforma selecionada para tal (Seq. i3). Em seguida, fez-se a execução da versão paralela para CPU utilizando a mesma plataforma (Par. i3). Na sequência, executou-se a versão paralela para GPU compartilhada nas duas GPUs selecionadas para tal (Par. GT 620 e Par. GTX 560). Por fim, executou-se a versão paralela para GPU dedicada nas GPUs selecionada para este fim (Par. GTX 560 Ded.).

Cada versão do programa foi executada 10 vezes em cada uma das plataformas selecionadas. Uma visão dos tempos médios de execução do cálculo de volume para as três amostras, nas diferentes plataformas, pode ser observada nos Gráficos 2, 3 e 4 que serão apresentados e comentados na sequência.

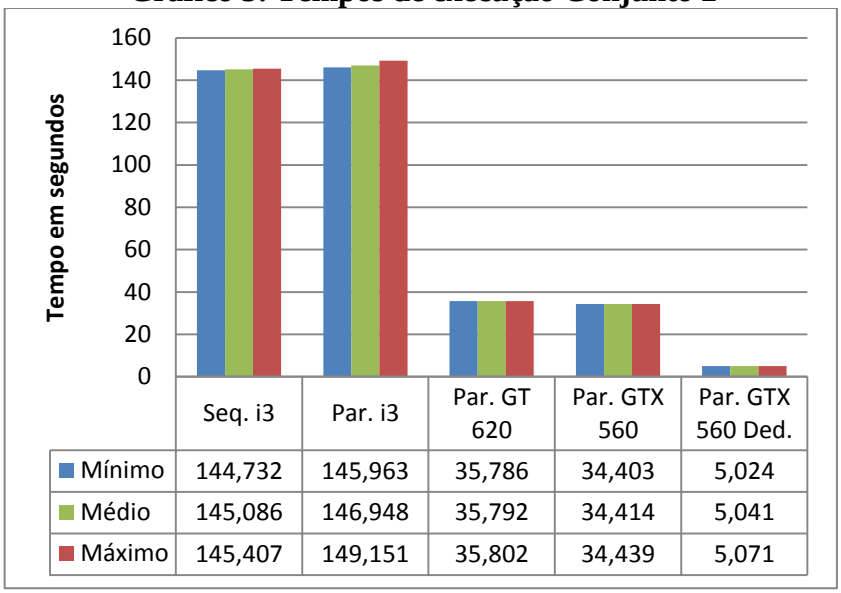
**Gráfico 2: Tempos de execução Conjunto 1**




| | Seq. i3 | Par. i3 | Par. GT 620 | Par. GTX 560 | Par. GTX 560 Ded. |
|---|---|---|---|---|---|
| Mínimo | 10,495 | 10,625 | 2,777 | 2,754 | 0,530 |
| Médio | 10,822 | 10,817 | 2,793 | 2,761 | 0,546 |
| Máximo | 11,419 | 11,127 | 2,808 | 2,778 | 0,592 |

**Fonte: Elaborado pelo autor**

**Gráfico 3: Tempos de execução Conjunto 2**




| | Seq. i3 | Par. i3 | Par. GT 620 | Par. GTX 560 | Par. GTX 560 Ded. |
|---|---|---|---|---|---|
| Mínimo | 144,732 | 145,963 | 35,786 | 34,403 | 5,024 |
| Médio | 145,086 | 146,948 | 35,792 | 34,414 | 5,041 |
| Máximo | 145,407 | 149,151 | 35,802 | 34,439 | 5,071 |

**Fonte: Elaborado pelo autor**

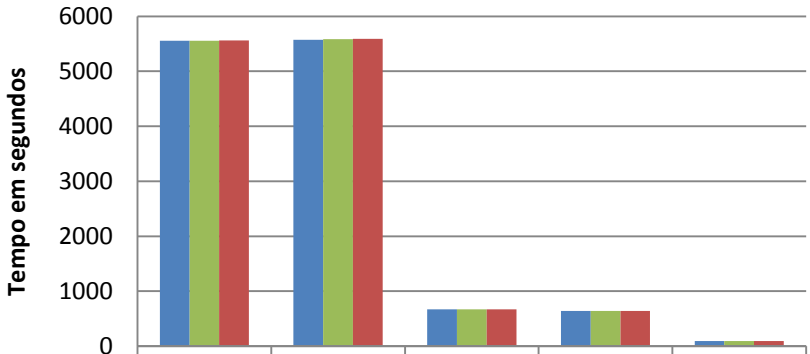
O Gráfico 2 apresenta o resumo dos tempos de execução para o conjunto amostral 1. Com ele, percebe-que houve reduções significativas quando do uso de GPU em relação ao uso de CPU e, além disso, entre as execuções sobre GPU, aquela feita utilizando-se uma GPU dedicada apresentou tempos de execução significativamente menores que as demais plataformas. Percebe-se ainda que é bem pequena a variação entre os valores mínimo, médio e máximo de tempo de execução da aplicação em cada plataforma, sobretudo naquelas baseadas em GPU, para as quais os três valores de medição são mais próximos entre si. Esta variação, por ser pequena, faz com que seja alta a probabilidade de que qualquer execução da

aplicação sobre determinada plataforma apresente tempos de execução próximos aos valores médios dos resultados experimentais encontrados para cada conjunto de dados.

O Gráfico 3 apresenta o resumo dos tempos de execução para o conjunto amostral 2. Com ele, assim como no Gráfico 2, percebe a tendência da diminuição dos tempos de execução quando do uso de GPU, sobretudo com a utilização de GPU dedicada. Observa-se aqui uma distribuição ainda mais equacionada dos valores mínimo, médio e máximo do tempo de execução, se comparados àqueles apresentados no Gráfico 2. Isso evidencia o bom comportamento da aplicação em manter seus tempos de execução próximos a um valor médio, sem apresentar grandes desvios de tal valor.

O Gráfico 4 apresenta o resumo dos tempos de execução para o conjunto amostral 3. Existe, assim como nos Gráficos 2 e 3, a tendência da diminuição dos tempos de execução quando do uso de GPU, principalmente com a utilização de GPU dedicada e, da mesma forma, é notado o bom comportamento da aplicação em manter seus tempos de execução bem próximos à média.

**Gráfico 4: Tempos de execução Conjunto 3**

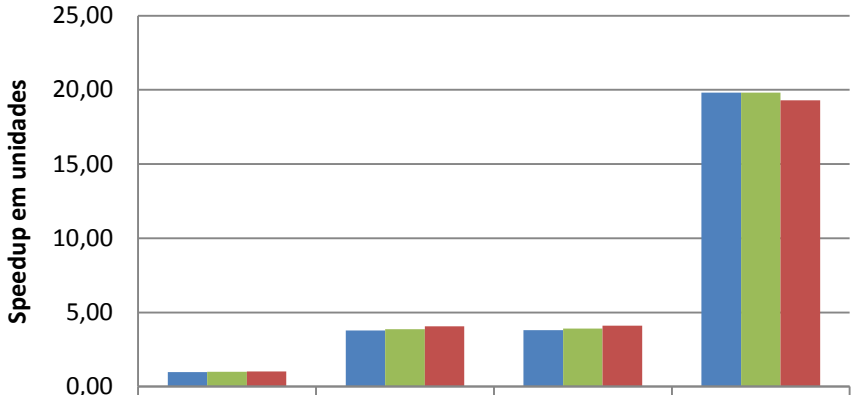



| | Seq. i3 | Par. i3 | Par. GT 620 | Par. GTX 560 | Par. GTX 560 Ded. |
|---|---|---|---|---|---|
| Mínimo | 5552,198 | 5574,499 | 667,993 | 638,795 | 92,950 |
| Médio | 5556,266 | 5582,751 | 668,186 | 641,647 | 92,974 |
| Máximo | 5560,334 | 5591,003 | 668,379 | 643,584 | 93,008 |

**Fonte: Elaborado pelo autor**

A partir dos valores de tempo de execução de cada experimento, fez-se uma análise do ganho de desempenho (*speedup*) dos programas. Os valores de *speedup* podem ser visualizados nos Gráficos 5, 6 e 7.

**Gráfico 5: *Speedup* para as execuções do Conjunto 1**

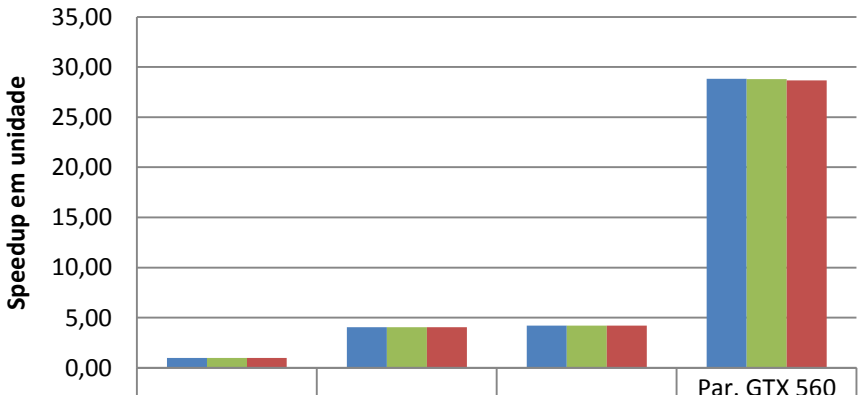



| | Par. i3 | Par. GT 620 | Par. GTX 560 | Par. GTX 560 Ded. |
|---|---|---|---|---|
| Mínimo | 0,99 | 3,78 | 3,81 | 19,80 |
| Médio | 1,00 | 3,87 | 3,92 | 19,82 |
| Máximo | 1,03 | 4,07 | 4,11 | 19,29 |

**Fonte: Elaborado pelo autor**

Com Gráfico 5 é possível observar os ganhos de desempenho para a execução da aplicação sobre o conjunto de dados 1. Nele observa-se a tendência do *speedup* em aumentar, quando da utilização de GPU em relação ao uso de CPU, sendo que a plataforma constituída de uma GPU dedicada é aquela que apresenta o maior ganho. Observa-se também a pequena variação entre os ganhos mínimo, médio e máximo, o que demonstra a estabilidade da aplicação no que se refere ao tempo execução para um determinada carga de dados.

**Gráfico 6: *Speedup* para as execuções do Conjunto 2**



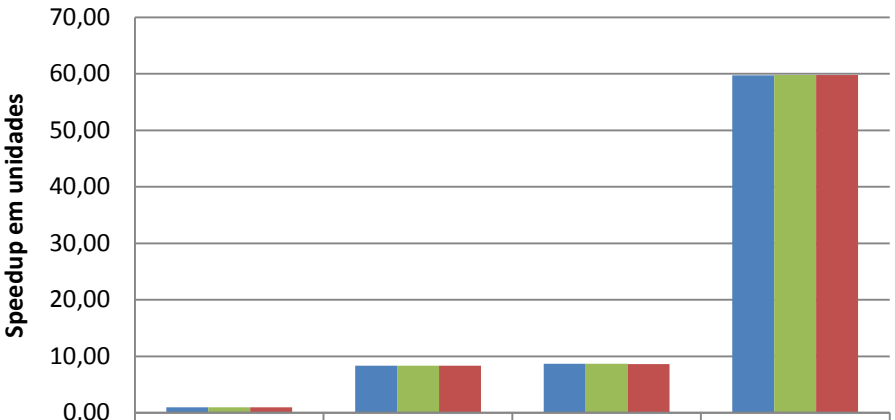

| | Par. i3 | Par. GT 620 | Par. GTX 560 | Par. GTX 560 Ded. |
|---|---|---|---|---|
| Mínimo | 0,99 | 4,04 | 4,21 | 28,81 |
| Médio | 0,99 | 4,05 | 4,22 | 28,78 |
| Máximo | 0,97 | 4,06 | 4,22 | 28,67 |

**Fonte: Elaborado pelo autor**

O Gráfico 6 apresenta os ganhos de desempenho para a execução da aplicação sobre o conjunto de dados 2 nas diferentes plataformas. Assim como no Gráfico 5, é bastante visível a tendência do *speedup* em aumentar, quando da utilização de GPU em relação ao uso de CPU, sobretudo com a GPU dedicada. No entanto, para essa amostra de dados, a variação entre os ganhos mínimos, médios e máximos é ainda menor.

O Gráfico 7 apresenta, enfim, os ganhos de desempenho obtidos na execução do conjunto de dados 3. Ele apresenta, da mesma forma que os anteriores, a tendência do *speedup* em aumentar, quando da utilização de GPU. Para o conjunto de dados 3 existe, no entanto, a variação entre os ganhos mínimos, médios e máximos é ainda menor que as execuções anteriores.

**Gráfico 7: *Speedup* para as execuções do Conjunto 3**




| | Par. i3 | Par. GT 620 | Par. GTX 560 | Par. GTX 560 Ded. |
|---|---|---|---|---|
| Mínimo | 1,00 | 8,31 | 8,69 | 59,73 |
| Médio | 1,00 | 8,32 | 8,66 | 59,76 |
| Máximo | 0,99 | 8,32 | 8,64 | 59,78 |

**Fonte: Elaborado pelo autor**

Com os resultados obtidos, foram feitas algumas análises acerca dos pontos mais significativos observados nos resultados e tal análise é apresentada na seção 5.5 a seguir.

## 5.5 Análise dos resultados

Os resultados mostram, de maneira geral, que as execuções em CPU, tanto sequencial quanto paralela, tiveram tempos de respostas aproximadamente iguais, com variações proporcionalmente irrelevantes. Por outro lado, as execuções em GPU apresentaram tempos de resposta diferentes entre si, prevalecendo os menores tempos para a GTX 560 em relação à GT 620, mas ambos significantemente menores que as execuções em CPU.

O *speedup* máximo alcançado com a execução em CPU foi de 1,03. Por outro lado, obteve-se a medida expressiva de 59,78 como *speedup* máximo para a GPU GTX 560. Com a utilização da CPU, os tempos de resposta mantiveram-se praticamente iguais, com variações proporcionalmente insignificantes no tempo de resposta das versões sequencial e paralela, o que pode ser evidenciado com os valores de *speedup* iguais a um ou bem próximos desse valor. Já com a utilização das GPUs, significativos ganhos de desempenho foram observados, o que vai ao encontro dos objetivos do presente trabalho.

Uma importante análise a ser feita refere-se à consideração da completude de pontos coletados para a efetuação dos cálculos. Durante a experimentação, notou-se que a quantidade de pontos da amostra interfere na precisão dos resultados, ou seja, os valores encontrados podem ser menos precisos caso se utilize apenas uma parte de uma determinada coleta.

Portanto, o ideal é que, para cada análise, sejam submetidos à aplicação do Método Kernel todos os pontos coletados que se referem a tal análise. Os resultados da aplicação construída mostram que, mesmo ao se utilizar uma carga de pontos significativamente maior, os tempos de resposta mantiveram-se proporcionalmente pequenos para as execuções em GPU, sobretudo na GTX 560. Para evidenciar a afirmativa anterior, tem-se que o pior tempo de execução na GPU GTX 560 para o Conjunto 2 (805 pontos, uma matriz de 5.198.598 células, processados em 5,701 segundos) é ainda menor que a execução mais rápida em CPU para o Conjunto 1 (67 pontos, 4.660.812 células, processados em 10,495 segundos); tem-se ainda que o pior tempo de execução na GPU GTX 560 para o Conjunto 3 (2.607 pontos, uma matriz de 62.922.860 células, processados em 93,008 segundos) é também menor que a execução mais rápida em CPU para o Conjunto 2 (805 pontos, uma matriz de 5.198.598 células, processados em 144,732 segundos). Ambos os casos devem considerar o número de cálculos de contribuição, que cresce bastante com o aumento do número de pontos e com o aumento do tamanho da matriz tridimensional. Os casos apresentados são evidenciados na Tabela 4, nas células coloridas de mesma cor.

**Tabela 4: Conjuntos de dados utilizados**

| | Pontos | Células | Número de cálculos de contribuição | Menor tempo em CPU | Maior tempo em GPU Ded. |
|---|---|---|---|---|---|
| Conjunto 01 | 67 | 4.660.812 | 312.274.404,00 | 10,495 | 0,592 |
| Conjunto 02 | 805 | 5.198.598 | 4.184.871.390,00 | 144,732 | 5,071 |
| Conjunto 03 | 2.607 | 62.922.860 | 164.039.896.020,00 | 5552,198 | 93,008 |

**Fonte: Elaborada pelo autor**

Os valores apresentados na tabela acima evidenciam que o uso de computação paralela com GPUs pode trazer ganhos significativos de desempenho e, ao mesmo tempo, consideráveis ganhos na qualidade dos resultados das aplicações de Visualização Científica, especificamente, pela possibilidade de se considerar, em tempo hábil, amostras de tamanhos maiores, uma vez que amostras maiores possibilitam resultados mais precisos, o que é de fundamental importância.

## 5.6 Considerações sobre os resultados

A aplicação construída permitiu a obtenção de informações acerca do uso do espaço tridimensional de uma forma inovadora. As informações sobre o volume e as imagens que o representam puderam ser geradas pela aplicação em tempo hábil, e com qualidade significativa aos pesquisadores do PPGZV.

As imagens geradas para representação da distribuição dos animais no ambiente tridimensional, bem como as imagens geradas para representação da área de vida utilizada e os resultados numéricos calculados pelo sistema sobre as representações construídas, servem como base para que os pesquisadores possam compreender melhor o padrão da utilização da área de vida pelos animais estudados. Portanto, evidencia-se que a Visualização Científica facilita o entendimento e a extração de informação e conhecimento de um conjunto de dados, quer sejam de simulação, experimentação ou observação, através da apresentação visual de tal conteúdo.

Do ponto de vista computacional, os experimentos apontaram vantagens significativas do uso de arquiteturas paralelas baseadas em GPU para fornecimento de capacidade computacional para aplicações que manipulem grandes e complexas massas de dados e/ou utilizem algoritmos complexos.

# 6 CONCLUSÕES

A Visualização Científica facilita o entendimento e a extração de informação e conhecimento de um conjunto de dados, quer sejam de simulação, experimentação ou observação, através da apresentação visual dos dados em estudo.

Os resultados iniciais do Projeto Guigó, apresentados no trabalho de Bonde (2011), evidenciam a afirmativa anterior. As imagens geradas para representação da distribuição da área de vida dos animais no ambiente bidimensional e tridimensional, bem como os resultados numéricos calculados pelo sistema sobre as representações construídas, serviram de grande auxílio para que a pesquisadora pudesse fomentar sua pesquisa e alcançar os objetivos propostos, dentre os quais listava-se a descoberta de informações sobre a utilização da área de vida pelos animais para uma compreensão mais efetiva de como os referidos animais usam o ambiente em que vivem.

No entanto, para que visualizações de boa qualidade fossem criadas em tempo hábil, uma grande capacidade computacional fez-se necessária. Utilizou-se, então, computação paralela baseada em GPU com CUDA para prover capacidade computacional necessária à aplicação de visualização construída. Os resultados mostraram ganhos significativos quando do uso de tais tecnologias, o que permitiu a execução da aplicação em tempo hábil, e com a qualidade necessária aos pesquisadores em Zoologia.

O uso de GPU como elemento de processamento de propósito geral mostrou-se, para o contexto desta pesquisa, uma alternativa viável para a disponibilização de capacidade de processamento necessária em aplicações complexas. Conseguiu-se reduzir os tempos de resposta das aplicações em até 59,78 vezes e ainda manter a qualidade necessária dos resultados, com uso de um componente significativamente mais barato em relação ao uso de soluções clusterizadas, ou baseadas em computadores de maior porte e que, além disso, possui consumo de energia menor que estas outras soluções.

Com tais reduções no tempo de execução das aplicações, obteve-se um software utilizável pelos pesquisadores em Zoologia dos Vertebrados, o que permite a complementação dos objetivos futuros propostos em Bonde (2011), abordados no estudo de caso desta pesquisa.

O que a presente pesquisa pode comungar com o trabalho de Howison; Bethel e Childs (2011) é o fato de que cada plataforma paralela possui características específicas, as quais precisam ser consideradas, por contribuírem para o desempenho das aplicações. Além disso,

mostrou-se que alternativas que possibilitem menos comunicação entre os nós de processamento são mais prováveis de alcançarem maiores ganhos de desempenho.

## 6.1 Principais contribuições

As principais contribuições do presente trabalho foram, portanto, relacionadas à aplicação dos conhecimentos da área de Computação na área de Zoologia. Entre as principais, citam-se: o modelo matemático adaptado ao cálculo da utilização do espaço de vida em três dimensões; os algoritmos sequenciais e paralelos construídos com base no modelo matemático, a aplicação que implementa tais algoritmos de cálculo e a apresentação simplificada dos resultados na forma de imagem, em atendimento às demandas do PPGZV. Além disso, não menos importante, citam-se os resultados do estudo experimental que evidenciam os ganhos de desempenho da utilização de GPUs como meio de prover capacidade computacional para aplicações complexas, no escopo específico de aplicações de Visualização Científica tridimensional.

## 6.2 Trabalhos futuros

A aplicação construída atendeu os propósitos definidos para o estudo de caso desta pesquisa, que eram o cálculo e apresentação da distribuição do uso do espaço de vida dos animais estudados, e possibilitou demonstrar o potencial do uso de GPUs em Visualização Científica. No entanto, alguns trabalhos são vislumbrados como forma de evoluir a pesquisa.

Propõe-se estudar e aplicar técnicas mais avançadas de computação gráfica para tornar a apresentação dos resultados dos cálculos mais fluida e interativa, com a manipulação das imagens de forma mais leve e natural, com a utilização mais eficiente do espaço da janela de visualização, técnicas avançadas de utilização de cores, como semiologia óptica e colorimetria, com a geração de vídeos sobre o uso da área de vida e o comportamento dos animais variando no tempo, entre outros. Propõe-se também adaptar a aplicação para que ela se torne genérica a ponto de que seja possível utilizá-la no estudo da área de vida de qualquer animal que utilize o espaço em três dimensões.

Um trabalho, também muito significativo, seria a implementação da aplicação com base em outras plataformas de computação paralela, como OpenCL ou OpenACC, para analisar o comportamento da aplicação em relação ao desempenho, à qualidade dos resultados e à facilidade de desenvolvimento, quando da utilização dessas tecnologias.

É importante também, como forma de evoluir a aplicação construída, propor e testar novos algoritmos paralelos que tornem mais eficientes, por exemplo, a divisão de trabalho entre os núcleos de processamento e o uso dos recursos computacionais disponibilizados.

Objetiva-se, ainda, aplicar as tecnologias utilizadas neste trabalho, tanto aquelas já existentes quanto as propostas aqui, em outros contextos da Visualização Científica, como em visualização da dinâmica de nuvens e campos eletromagnéticos, entre outros.

Por fim, propõe-se definir e implementar mecanismos de tomada de decisão que tornem a aplicação inteligente para escolher a forma como vai dividir o trabalho a ser realizado. Incluem-se nesta escolha a definição do número de *threads*, da quantidade de blocos de *threads*, a seleção da porção de trabalho enviada para a GPU de cada vez, a escolha do uso dos diferentes tipos de memória da placa gráfica, entre outros elementos, que podem desencadear uma utilização mais efetiva dos recursos disponíveis.

## REFERÊNCIAS

AMD – ADVANCED MICRO DEVICES. **The world's irst combination of low-power CPU and advanced GPU integrated into a single embedded device**. Disponível em: http://www.amd.com/br/Documents/49282B_Embedded_Solutions_GSeries_Brief.pdf. Acessado em: 19 de outubro de 2011.

AMDAHL, Gene. **Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities.** *AFIPS Conference Proceedings* (30): 483–485. Disponível em: http://www-inst.eecs.berkeley.edu/~n252/paper/Amdahl.pdf. Acessado em: 02 de julho de 2013.

AZEVEDO, Eduardo; CONCI, Aura. **Computação Gráfica**: Teoria e Prática. Rio de Janeiro. Elsevier. 2003. 353 p.

BARNEY, Blaise. **OpenMP**. Lawrence Livermore National Laboratory. Disponível em: < https://computing.llnl.gov/tutorials/openMP/>. Acessado em: 19 de outubro de 2011.

BONDE, Marina. **Modelando o habitat em três dimensões**: Callicebus nigrifrons (Spix, 1823) como modelo. 2011. Dissertação (Mestrado) – Pontifícia Universidade Católica de Minas Gerais, Programa de Pós-Graduação em Zoologia dos Vertebrados, Belo Horizonte. BR-MG.

CHEN, GuoLiang et al. **Integrated research of parallel computing**: Status and future. Science China Press, co-published with Springer, jun 211, p.1845-1853. Disponível em: http://dx.doi.org/10.1007/s11434-009-0261-9.

DEBE, Eric. **Recent changes and future trends in general purpose processor architectures to support image and video applications**. International Conference on Image Processing, ICIP 2003, Proceedings, vol.3, set. 2003, p. III-85. Disponível em: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1247187&isnumber=27938.

DONGARRA, Jack. **Trends in high performance computing**: a historical overview and examination of future developments. Circuits and Devices Magazine, IEEE, vol. 22, n. 1, p 22-27, jan-fev. 2006. Disponível em: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1598076&isnumber=33612.

ELVINS, Thomas Todd. **A Survey of Algorithms for Volume Visualization**. ACM Computer Graphics, vol. 26, n. 3, aug. 1992. p. 194–201. Disponível em: http://dl.acm.org/citation.cfm?id=142427.

FERREIRA, Aurélio Buarque de Holanda. **Aurélio Século XXI**: o dicionário da Língua Portuguesa. 3ª ed. Rio de janeiro. Nova Fronteira. 1999. 2128 p.

FISHER, Robert. **Kernel Density Estimators**. (Tutorial). Disponível em: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0405/MISHRA/kde.html. Acessado em 14 de março de 2012.

FLYNN, Michael J. **Very high-speed computing systems**. Proceedings of the IEEE, vol. 54, n. 12, dez. 1996, p. 1901-1909. Disponível em: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1447203&isnumber=31091.

FLYNN, Michael J. **Some Computer Organizations and Their Effectiveness**. IEEE Transactions on Computers, vol. C-21, n.9, set. 1972, p. 948-960. Disponível em: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5009071&isnumber=5009065.

FOLEY, James David et al. **Computer Graphics:** Principles and Practice. 2ª ed. em C. Boston. Addison-Wesley. 1996. 1175p.

FUCHS, R.; HAUSER, H. **Visualization of Multi-Variate Scientific Data**. Comput Graphics Forum, Proceedings, vol. 28, n. 6, 2009, p. 1670-1690. Disponível em: http://www.inf.ethz.ch/personal/rafuchs/pdf/cgf_star.pdf.

HAGEN, Hans et al. **Scientific Visualization**: methods and applications. Proceedings of the 19th spring conference on Computer graphics (SCCG '03), ACM, New York, p 23-33. 2003. Disponível em: http://doi.acm.org/10.1145/984952.984957.

HANSEN, Charles D.; JOHNSON, Chris R. **The Visualization Handbook**. Amsterdam. Elsevier Inc. 2005. 984 p.

HOWISON, M.; BETHEL, E.; CHILDS, H. **Hybrid Parallelism for Volume Rendering on Large, Multi- and Many-core Systems**. Visualization and Computer Graphics, IEEE Transactions on, vol. 18, n. 1, jan. 2011, p. 1. Disponível em: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5708139&isnumber=4359476.

HU-BERLIM – UNIVERSIDADE HUMBOLDT DE BERLIM. **Multivariate Kernel Density Estimation**. (Tutorial). Disponível em: http://sfb649.wiwi.hu-berlin.de/fedc_homepage/xplore/ebooks/html/spm/spmhtmlnode18.html. Acessado em 14 de março de 2012. (Acessado através do RDC - Research Data Center - da Escola de Negócios e Economia da Universidade Humboldt de Berlim).

INTEL CORPORATION. **Tecnologia Multi-Core Intel®**. Disponível em: http://www.intel.com/portugues/multi-core/index.htm. Acessado em: 19 de outubro de 2011.

INTEL CORPORATION. **Intel® Hyper-Threading Technology**. Disponível em: http://www.intel.com/content/www/us/en/architecture-and-technology/hyper-threading/hyper-threading-technology.html. Acessado em: 19 de outubro de 2011.

INTEL CORPORATION. **2nd Generation Intel® Core™ Processor Family: Product Brief**. Disponível em: http://www.intel.com.br/content/www/br/pt/processors/core/2nd-gen-core-desktops-brief.html. Acessado em: 19 de outubro de 2011.

JOHN, Savage E. **Models of Computation**: Exploring the Power of Computing. Boston. Addison-Wesley Longman. 1997. 698 p.

KHRONOS GROUP**. OpenGL Overview.** Disponível em: www.opengl.org/about. Acessado em 16 de maio de 2013.

KINZEY, W. G; COIMBRA-FILHO, A.F. (ed.); Mittermeier R.A. (ed.). **Ecology and Behaviour of Neotropical Primates**. Rio de Janeiro. Academia Brasileira de Ciências. 496 p.

LACROUTE, Philippe; LEVOY, Marc. **Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation**. SIGGRAPH – International Conference on Computer Graphics and Interactive Techniques, ACM – Association for Computing Machinery. [S.l.]. 1994. p. 451–458.

LEVOY, Marc. **Display of Surfaces from Volume Data**. IEEE Computer Graphics & Applications, vol. 8, n. 3, mai. 1988, p 29–37. Disponível em: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=511&isnumber=31.

LEVOY, Marc. **Efficient Ray-Tracing of Volume Data**. ACM Transactions on Graphics, vol. 9, n. 3, jul. 1990, p. 245–261. Disponível em: http://dl.acm.org/citation.cfm?id=78965.

MACEDONIA, Michael; **The GPU enters computing's mainstream.** IEEE Computer, vol. 36, n. 10, out. 2003, p. 106-108. Disponível em: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1236476&isnumber=27718.

MANDL, Peter; BORDOLOI, Udeepta. **General-purpose Graphics Processing Units Deliver New Capabilities to the Embedded Market**. Disponível em: http://www.amd.com/br/Documents/GPGPU-Embedded.pdf. Acessado em: 19 de outubro de 2011.

MANSSOUR, Isabel Harb; FREITAS, Carla Maria Dal Sasso. **Visualização Volumétrica**. RITA – Revista de Informática Teórica e Aplicada, Instituto de Informática, Universidade Federal do Rio Grande do Sul, vol. 9, n. 2, 2002.

MATRIX LABORATORY – MATLAB. **MATLAB - The Language Of Technical Computing**. Disponível em: http://www.mathworks.com/products/matlab/?s_cid=global_nav. Acessado em: 08 de novembro de 2011.

MAX, Nelson. **Progress in Scientific Visualization**. Journal The Visual Computer, Springer Berlin / Heidelberg, vol. 21, n. 12, p. 979 – 984. Disponível em: http://dx.doi.org/10.1007/s00371-005-0361-8.

MCCORMICK, B. H.; DEFANTI, T. A.; BROWN, M. D. **Visualization in Scientific Computing**. Computer Graphics and Applications, IEEE. vol.7, n.10, p. 69, out. 1987. Disponível em: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4057039&isnumber=4057022.

MONOLEY, B. et al. **Sort-First Parallel Volume Rendering**. Visualization and Computer Graphics, IEEE Transactions on, vol.17, n.8, ago. 2011, p.1164-1177. Disponível em: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5582082&isnumber=5872084.

NVIDIA CORPORATION. **O que é CUDA?** Disponível em: http://www.nvidia.com.br/object/cuda_home_new_br.html. Acessado em 18 de junho de 2012.

NVIDIA CORPORATION. **CUDA C Programming Guide.** Disponível em: http://docs.nvidia.com/cuda/cuda-c-programming-guide. Acessado em 01 de julho de 2013.

NVIDIA CORPORATION. **NVIDIA GeForce GT 620.** Disponível em: http://www.nvidia.com.br/object/geforce-gt-620-br.html. Acessado em 01 de julho de 2013.

NVIDIA CORPORATION. **NVIDIA GeForce GTX 560.** Disponível em: http://www.nvidia.com.br/object/product-geforce-gtx-560-br.html. Acessado em 01 de julho de 2013.

OLIVEIRA, R. C. R.; COELHO, A. S.; MELLO, F. R. **Estimativa de densidade e tamanho populacional de sauá (Callicebus nigrifrons) em um fragmento de mata em regeneração**, **Viçosa, Minas Gerais, Brasil**. Neotropical Primates, vol.11, n. 2, ago. 2003, p.91–94. Disponível em: http://www.primate-sg.org/PDF/NP11.2.nigrifrons.populacao.pdf.

PAIVA, Anselmo Cardoso; SEIXAS, Roberto de Beauclair; GATTASS, Marcelo. **Introdução à Visualização Volumétrica**. 2011. 107f. Disponível em: http://www.dbd.puc-rio.br/depto_informatica/99_03_paiva.pdf. Acessado em: 08 de novembro de 2011.

PITAC – PRESIDENT'S INFORMATION TECHNOLOGY ADVISORY COMMITTEE. **Computational Science: Ensuring America´s Competitiveness**. PITAC Report to the President, EUA, jun. 2005. Disponível em http://www.nitrd.gov/pitac/reports/20050609_computational/computational.pdf.

ROSE, Jonathan et al. **Architecture of field-programmable gate arrays**: the effect of logic block functionality on area efficiency. Solid-State Circuits, IEEE Journal of , vol. 25, n. 5, out. 1990, p.1217-1225. Disponível em: URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=62145&isnumber=2262.

SBC – SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. **Grandes Desafios da Pesquisa em Computação no Brasil – 2006 – 2016**. Relatório Sobre o Seminário Grandes Desafios da Pesquisa em Computação no Brasil – 2006 – 2016, Brasil, mai. 2006. Disponível em: http://www.sbc.org.br/index.php?language=1&content=downloads&id=272.

SONG, Iang. **Nvidia Graphics Processing Unit (GPU).** Disponível em http://www2.engr.arizona.edu/~yangsong/gpu.htm. Acessado em 28 de junho de 2013.

Udupa, J.K.; Odhner, D. **Shell Rendering**. IEEE Computer Graphics and Applications, vol. 13, n. 6, p. 58-67, nov. 1993. Disponível em: http://dx.doi.org/10.1109/38.252558.

UPSON, Craig; KEELER, Michael. **V-buffer: visible volume rendering**. ACM SIGGRAPH Computer Graphics, vol. 22, n. 4, ago. 1988, p. 59-64. Disponível em: http://doi.acm.org/10.1145/54852.378482.

VAN PELT, R.; VILANOVA, A.; VAN DE WETERING, H. **Illustrative Volume Visualization Using GPU-Based Particle Systems**. Visualization and Computer Graphics, IEEE Transactions on, vol. 16, n. 4, jul-ago. 2010. Disponível em: URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5416701&isnumber=5465868.

VARDI, Moshe Y. **Science has only two legs**. *Magazine Communications of the ACM*, vol. 53, n. 9, set. 2010. Disponível em: http://doi.acm.org/10.1145/1810891.1810892.

WAND, M. P.; Jone, M. C. **Kernel Smoothing.** Monographs on Statistics and Applied Probability. Chapman & Hall, 1995.

WESTERMANN, R. **Parallel volume rendering**. Parallel Processing Symposium, 1995. Proceedings., 9th International , abr. 1995, p.693-699. Disponível em: URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=395873&isnumber=8974.

WESTOVER, Lee. **Interactive Volume Rendering**. Workshop on Volume Visualization. University of North Carolina, Chapel Hill. 1989. p. 9–16.

WESTOVER, Lee. **Footprint Evaluation for Volume Rendering**. SIGGRAPH – International Conference on Computer Graphics and Interactive Techniques, ACM – Association for Computing Machinery. [S.l.]. 1990. p. 367–376.

WIKIPEDIA (EN), the free encyclopedia. **Scientific Visualization**. Disponível em: http://en.wikipedia.org/wiki/Scientific_visualization. Acessado em: 11 de novembro de 2011.

WIKIPEDIA (PT), a enciclopédia livre. **Multinúcleo**. Disponível em: http://pt.wikipedia.org/wiki/Multinúcleo. Acessado em: 19 de outubro de 2011.

WIKIPEDIA (PT), a enciclopédia livre. **Unidade de Processamento Gráfico**. Disponível em: http://pt.wikipedia.org/wiki/Unidade_de_processamento_gráfico. Acessado em: 19 de outubro de 2011.

WILHELMS, Jane; VAN GELDER, Allen. **A Coherent Projection Approach for Direct Volume Rendering**. ACM SIGGRAPH Computer Graphics, vol. 25, n. 4, p. 275-284, jul. 1991. Disponível em: http://dl.acm.org/citation.cfm?id=122718.122758.

WOOD, Derick. **Theory of Computation**. New York. Jhon Wiley & Sons. 1987. 576 p.

WORTON, B. J. **Kernel Methods for Estimating the Utilization Distribution in Home-Range Studies**. Ecology, Ecological Society of America, vol. 70, n 1, p 164 – 168, fev. 1989. Disponível em: http://www.jstor.org/stable/1938423.

YAGEL, Roni; COHEN, Daniel; KAUFMAN, Arie. **Discrete Ray Tracing**. IEEE Computer Graphics and Applications, vol. 12, n. 5, set. 1992, p. 19-28. Disponível em: http://dx.doi.org/10.1109/38.156009.