Aceleração de Ray Tracing via Heterogeneidade *

Wagner Toledo Corrêa
Maurício Antônio de Castro Lima
Wagner Meira Junior
Márcio Luiz Bunte de Carvalho

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Caixa Postal 702
30161-970 Belo Horizonte, MG, Brasil
wagner@dcc.ufmg.br

Abstract. Ray tracing is a simple but powerful image synthesis technique known to produce realistic images but at a high computational cost and therefore with long computing execution times. Several acceleration techniques have already been devised to minimize this drawback. This work presents two acceleration techniques which take advantage over the inherent parallelism of ray tracing. The first technique uses a MIMD machine and the second one uses an heterogeneous parallel system consisting of a MIMD and a SIMD machine. The aim of this work is to prove that heterogeneity makes the second more efficient than the first. The results show that the heterogeneous technique produces good savings in computing execution times in comparison to the homogeneous one.

Introdução

Duas características principais fazem com que *ray tracing* seja uma das técnicas de síntese de imagens mais conhecidas: o alto grau de realismo das imagens geradas e o elevado custo computacional envolvido no processo de produzi-las. Com o objetivo de otimizar o processo de *ray tracing*, várias técnicas de aceleração já foram desenvolvidas. Este trabalho apresenta duas técnicas que utilizam a execução concorrente de operações como forma de acelerar *ray tracing*. A primeira técnica utiliza uma máquina do tipo SIMD e a segunda utiliza um sistema paralelo heterogêneo que consiste de uma máquina do tipo MIMD e outra do tipo SIMD. O objetivo deste trabalho é provar que a heterogeneidade torna a segunda técnica mais eficiente do que a primeira.

O texto está divido em 6 seções. Na seção 1, é realizada uma breve descrição do algoritmo clássico de *ray tracing*. Na seção 2, as técnicas de aceleração mais conhecidas são abordadas sucintamente. As duas técnicas de aceleração de *ray tracing* por paralelismo são apresentadas na seção 3. O ambiente particular em que as técnicas foram implementadas é descrito na seção 4. Os resultados que comprovam a superioridade da técnica heterogênea com relação à técnica homogênea encontram-se na seção 5. Por fim, uma análise da técnica heterogênea é feita na seção 6.

*Trabalho financiado com recursos da FAPEMIG (proc. TEC 1113/90) e do CNPq (proc. 502353/91-0(NV)).

1 O algoritmo clássico de *ray tracing*

1.1 Descrição geométrica

O algoritmo clássico de *ray tracing* foi proposto em 1980 por Whitted [Whitted, 1980]. Para entendê-lo, pode-se imaginar que um observador está olhando para uma cena através da tela do computador e que raios de luz provenientes da cena atingem o olho do observador passando pelos centros dos *pixels*. A cor de um *pixel* será a cor do raio proveniente da cena que passa pelo seu centro. Como nem todo raio proveniente da cena atinge o observador, na realidade, o processo é feito em sentido inverso. Para cada *pixel* da tela, lança-se um "raio primário", passando pelo seu centro, partindo do olho do observador em direção à cena. Determina-se qual objeto da cena o raio atinge primeiro e em que ponto ocorre a intersecção. Levando-se em conta as fontes de luz presentes na cena, um modelo de iluminação é aplicado ao ponto de intersecção. As diversas componentes da luz são calculadas e somadas. A cor resultante é a cor do *pixel*.

Normalmente, um modelo de iluminação determina a quantidade de luz que chega em um ponto somando três componentes básicas. A primeira é a contribuição local que depende apenas de fontes de luz reais. A segunda componente é a quantidade

de luz que atinge o ponto devido à reflexão da luz por um objeto brilhante. A última contribuição é a quantidade de luz que é transmitida por refração através de um objeto transparente. Cada uma dessas duas últimas componentes é obtida pelo lançamento de "raios secundários" para calcular suas três subcomponentes (local, refletida e transmitida) e assim sucessivamente, criando uma "árvore de raios".

Como as direções dos raios secundários gerados por reflexão ou refração são calculadas levando em conta as leis físicas da ótica geométrica, as imagens geradas apresentam um alto grau de realismo, contendo efeitos visuais interessantes produzidos por objetos como espelhos, lentes e aquários. Podem existir reflexões múltiplas em que um raio é refletido por vários objetos brilhantes antes de atingir o observador e combinações complexas de reflexão, refração e sombras.

1.2 *Ray tracing* exaustivo

A principal tarefa em *ray tracing* é determinar o ponto de intersecção de um raio com um objeto. É fato conhecido que, numa cena típica, o custo dos cálculos de intersecção é bem maior do que a soma dos custos das tarefas restantes como cálculos de sombreamento. De acordo com Whitted [Whitted, 1980], esse custo pode chegar a representar mais de 95% do custo total da geração da imagem para cenas complexas.

A implementação mais direta do algoritmo de *ray tracing* é conhecida pelo nome de "*ray tracing* exaustivo". Esse nome é devido ao fato de que, para determinar o ponto de intersecção mais próximo ao observador, cada raio lançado é interseccionado com todos os objetos da cena. O excessivo número de cálculos de intersecção torna o *ray tracing* exaustivo muito inefeciente. A seguir, são apresentadas algumas das principais técnicas cujo objetivo é otimizar *ray tracing*.

2 Técnicas de aceleração

A maioria das técnicas de aceleração de *ray tracing* concentra-se no problema do cálculo de intersecções e assume que apenas uma parte desprezível do tempo é gasta nas demais tarefas. Basicamente, existem quatro estratégias para acelerar o processo de *ray tracing*:

Realizar intersecções mais rapidamente:
Consiste em reduzir o custo médio de intersecionar um raio com a cena. Pode-se conseguir isso de duas maneiras: utilizando algoritmos eficientes para objetos primitivos específicos ou diminuindo o número de intersecções de raios com objetos a serem feitas. Algoritmos que lidam com objetos específicos fazem com que a técnica de aceleração seja indesejavelmente dependente da definição do objeto [Hanrahan, 1983; Sederberg and Anderson, 1984]. Por isso, é melhor tentar diminuir o número de intersecções de raios com objetos. Dentre as formas de se fazer isso, destacam-se: hierarquia de volumes delimitadores [Whitted,1980; Rubbin and Whitted,1980; Gervautz, 1986; Roth, 1982; Kay and Kajiya, 1986; Goldsmith and Salmon, 1987], subdivisão espacial [Glassner, 1984; Fujimoto et al., 1986; Kaplan, 1985; Jansen, 1986;] e técnicas direcionais [Haines and Greenberg, 1986; Ohta and Maekawa, 1987; Arvo and Kirk, 1987].

Lançar menos raios:
Consiste de técnicas que reduzem o número de raios que precisam ser interseccionados com a cena, sejam os raios primários ou secundários. Um exemplo de tais técnicas é o "controle adaptativo de profundidade de árvore" [Hall and Greenberg, 1986]. Normalmente, a árvore de raios termina quando uma profundidade máxima predefinida é atingida ou superfícies opacas sao encontradas. Nessa técnica, ao contrário, a árvore de raios termina quando a contribuição de um raio para a cor de um *pixel* é menor do que um dado valor. Isso torna possível a eliminação do processamento de muitos raios de tal forma que a alteração no resultado final da imagem é imperceptível. Outra forma de eliminar muitos raios primários e, portanto, subárvores inteiras, é aplicar métodos estatísticos capazes de detectar as regiões da tela em que uma pequena amostragem de raios é suficiente para gerar resultados de boa qualidade [Cook, 1986; Kajiya, 1983; Lee et al., 1985; Purgathofer, 1986].

Utilizar entidades mais gerais:
Consiste em substituir o raio por uma entidade mais genérica que tem o raio comum como um caso particular. Os raios comuns permitem uma representação fácil, cálculos de intersecção eficientes e grande generalidade. Pode-se abrir mão dessas vantagens em troca de outros benefícios, como execução mais rápida, *anti-aliasing* efetivo e efeitos visuais adicionais. Isso é conseguido lidando simultaneamente com raios agrupados sob a forma de feixes [Heckbert and Hanrahan, 1984], cones [Amanatides, 1984] ou lápis [Shinya et al., 1987]. Entretanto, cada um desses "raios generalizados" apresenta alguma desvantagem como restrições na classe de objetos primitivos ou cálculos de intersecção aproxi-

mados.

Executar operações concorrentemente:
Consiste em aproveitar o paralelismo inerente ao processo de *ray tracing*, devido ao fato de que a cor de um *pixel* não depende da cor de outros *pixels*, isto é, os raios lançados para o interior da cena são independentes uns dos outros. Assim, o processamento de um raio pode ocorrer concorrentemente ao processamento de outros raios. Dentre as técnicas que tentam aproveitar essa característica de *ray tracing* estão vetorização [Max, 1981; Plunkett and Bailey, 1985], execução em uma rede de computadores de uso geral [Lima et al., 1993], execução em um multicomputador de uso geral [Goldsmith and Salmon, 1985] e execução em *hardware* específico [Ullner, 1983; Dippe and Wold, 1985; Nishimura et al., 1983; Kobayashi et al., 1987; Nemoto and Omachi, 1986; Clearly et. al., 1985].

3 Paralelização de *Ray tracing*

A primeira tentativa no sentido de aumentar a eficiência da síntese de imagens por *ray tracing* foi simplesmente distribuir as operações entre processadores de uma máquina MIMD. Entretanto, essa técnica revelou-se ser passível de otimização, o que motivou o desenvolvimento de uma técnica heterogênea que utiliza máquinas SIMD e MIMD. A seguir, cada uma dessas técnicas será explicada.

A técnica homogênea nada mais é do que um esquema do tipo mestre/escravo. Um processo mestre divide a imagem a ser gerada em regiões retangulares chamadas quadros e as distribui entre os processos escravos. Cada processo escravo executa os cálculos correspondentes ao quadro recebido e o devolve ao processo mestre. Numa situação ideal de paralelização, existiria um processo escravo para cada *pixel*. Na prática, isso é inviável, devido ao *overhead* de comunicação entre processadores resultante. Um grande problema dessa técnica, é a falta de um mecanismo para tratar as ocorrências dos chamados quadros degenerados (regiões da tela em que os raios primários não interseccionam nenhum objeto). Embora esses quadros possam ser completamente desconsiderados, o processo mestre não tem a capacidade de detectá-los antes de enviá-los aos processos escravos, causando um desperdício de tempo devido a cálculos e trocas de mensagens desnecessários.

Com o intuito de otimizar a técnica homogênea, investiu-se num algoritmo que utiliza uma máquina SIMD em conjunto com uma máquina MIMD, caracterizando, assim, um ambiente paralelo heterogêneo. A técnica heterogênea consiste de duas fases. Na primeira fase, o processo mestre transfere o controle para a máquina SIMD que realiza os cálculos referentes a todos os raios primários, simultaneamente. Na segunda fase, o processo mestre transfere o controle para a máquina MIMD que se encarrega de terminar a geração da imagem, tratando dos raios secundários. A otimização consiste no fato de que a máquina SIMD não processa os raios primários em quadros, mas todos de uma vez. Com esse mecanismo, os raios primários são tratados mais rapidamente e a máquina SIMD tem condições de informar a máquina MIMD das ocorrências de quadros degenerados, evitando que eles sejam distribuídos aos processos escravos, resolvendo esse problema da técnica homogênea. Como será visto na seção 5, foi obtido um ganho significativo em termos de eficiência através do uso dessa heurística.

Cabe ressaltar que, em ambas as técnicas, cada um dos processos escravos precisa ter uma cópia completa da estrutura de dados que armazena a definição da cena. Outro detalhe importante é que, através da paralelização, consegue-se uma redução nos tempos de execução devido ao simples fato de que os cálculos foram distribuídos entre vários processadores. Todavia, isso não quer dizer que houve uma diminuição nos custos computacionais, isto é, o número de cálculos de intersecção realizados é o mesmo. Também deve ser notado que o tamanho do quadro que minimiza os tempos de execução é um parâmetro a se determinar empiricamente, pois depende da arquitetura utilizada [Lima et al., 1993].

4 Ambiente de implementação

Para permitir uma comparação quantitativa entre as técnicas homogênea e heterogênea, um sistema de geração de imagens foi implementado. O usuário desse sistema modela a cena cuja imagem será confeccionada através de uma linguagem de descrição baseada nos conceitos de geometria sólida construtiva. Uma vez definida a cena, o usuário escolhe a forma pela qual a imagem será gerada: utilizando a técnica homogênea ou a heterogênea. A máquina SIMD utilizada na técnica heterogênea foi a *Zephyr Wavetracer* tratada na seção 4.1, enquanto que a máquina MIMD necessária em ambas as técnicas foi emulada por uma rede de *workstations*, utilizando um pacote para programação paralela denominado *PVM* apresentado na seção 4.2.

4.1 A Zephyr Wavetracer

4.1.1 Arquitetura

A Zephyr Wavetracer (WT) é uma máquina do tipo SIMD (Single Instruction Multiple Data) com 8192

processadores de 1 bit cada. Esses 8K processadores são dispostos em duas placas, cada uma com 4K processadores. Essa máquina é ligada a uma *workstation* hospedeira por uma interface SCSI.

A memória total da máquina (da ordem de 256 megabytes) é igualmente dividida entre os processadores. Cada processador possui dois tipos de memória. A primeira é da ordem de 2 Kbits e é interna ao processador. Já a segunda é externa a cada processador e possui 32 Kbytes.

Os processadores podem ser organizados bi ou tridimensionalmente via *software*. No caso de uma máquina com 8k processadores, o arranjo bidimensional tem dimensões 64 × 128 e o tridimensional 16 × 32 × 16.

Às dimensões do arranjo de processadores dá-se o nome de espaço de soluções. Num espaço de soluções de dimensões 64 × 128, podem ser manipulados simultânea e identicamente 8192 elementos.

4.1.2 Processamento Virtual

Para suportar um espaço de soluções com dimensões maiores que as dos arranjos de processadores, um controlador particiona a memória de cada processador em n partes iguais e associa cada uma delas a um nodo no espaço de soluções. Assim, sempre que uma instrução for gerada para o arranjo de processadores, o controlador instrui a sua execução para cada uma das partições via alterações de endereçamento, ou seja, o controlador trata cada processador real como n processadores virtuais.

4.1.3 Linguagem MultiC

Um programa de aplicação para a WT é escrito como um programa para a *workstation* hospedeira em MultiC, uma extensão da linguagem Ansi C. Tendo em vista que o programa é executado na *workstation* e não no controlador da WT, o programa de aplicação compilado tem acesso às facilidades da *workstation*, como sistema de arquivos, rede e ambientes gráficos. As dimensões do arranjo de processamento virtual são especificadas em tempo de execução, sem o prévio conhecimento da configuração do arranjo físico. Como resultado, um programa pode ser executado em máquinas com quaisquer quantidades de processadores sem necessidade de recompilação.

As principais extensões ao Ansi C providas pelo MultiC são as seguintes:

1. O especificador de tipos multi, que declara uma variável que tem um valor numérico independente em cada processador virtual. O especificador de tipo uni também foi acrescentado à linguagem com a finalidade de declarar variáveis que tenham apenas um valor que é armazenado na *workstation*.

2. Um operador de comunicação interprocessador que permite o trânsito e intercâmbio de dados entre processadores. Deve ser ressaltado que o fluxo de dados é idêntico para todo o espaço de soluções virtual.

3. Operadores de redução que aplicam a operação indicada a todos os valores da expressão multi alvo nos processadores virtuais ativos, condensando o resultado em um único tipo uni. As operações de redução são: soma, subtração, multiplicação, divisão, operações lógicas (e, ou, ou-exclusivo) e operadores de máximo e mínimo.

4.2 Arquitetura PVM

O PVM (*Parallel Virtual Machine*) [Geist et al., 1993] é um pacote de programas e bibliotecas que permite que uma rede de computadores heterogêneos UNIX seja usada como um único e grande computador paralelo. Assim, grandes problemas computacionais podem ser resolvidos usando o poder agregado de vários computadores.

Sob o PVM, o usuário define uma coleção de computadores seriais, paralelos e vetoriais que se comportarão como um único computador paralelo com memória distribuída. Desta forma, denominam-se máquina virtual esses computadores lógicos com memória distribuída e hospedeiro cada um dos computadores reais. O PVM provê funções para disparar tarefas automaticamente nas máquinas virtuais e permitir que as tarefas comuniquem e sincronizem entre si. Uma tarefa é definida como sendo a unidade de computação do PVM de forma análoga a processos UNIX. Freqüentemente ela é um processo UNIX, mas não necessariamente. Aplicações, que podem ser escritas em C ou FORTRAN 77, podem ser paralelizadas usando construções para trocas de mensagens comuns a muitos computadores de memória distribuída. Um bom exemplo de utilização do PVM é a implementação de aplicações do tipo mestre/escravo.

PVM suporta heterogeneidade nos níveis de aplicação, de máquina e de rede. Em outras palavras, ele permite que tarefas de aplicação explorem a arquitetura mais adequada à solução do problema enfocado. Todas as conversões de dados necessárias são automaticamente efetuadas para viabilizar a comunicação entre computadores que usem diferentes representações de inteiros e reais.

O sistema PVM é composto de duas partes. A primeira parte é um *daemon*, chamado *pvmd3* que reside em todos os computadores formando a máquina

virtual. O *pvmd3* foi projetado de tal forma que qualquer usuário com um *login* válido possa instalá-lo em uma máquina. Ao executar uma aplicação, o PVM executa o *pvmd3* em um dos computadores, disparando o *daemon* em cada um dos outros computadores e criando a máquina virtual definida pelo usuário. A aplicação PVM pode então ser disparada de qualquer dessas máquinas como uma tarefa UNIX normal. Usuários podem configurar máquinas virtuais com sobreposição, e cada um pode executar várias aplicações PVM simultaneamente.

A segunda parte do sistema é uma biblioteca de rotinas. Essa biblioteca contém rotinas que podem ser chamadas pelo usuário para troca de mensagens, disparo de processos, coordenação de tarefas e modificação da máquina virtual. Os programas de aplicação devem incluir essa biblioteca para usar o PVM.

5 Resultados

Nesta seção, é feita uma comparação quantitativa entre as técnicas homogênea e heterogênea. A massa de dados para testes consiste de cenas contendo esferas de tamanho e posição aleatórios. As técnicas foram comparadas com relação a três parâmetros: o número de objetos da cena, a área ocupada da imagem (dada pelo número de *pixels* para os quais os raios primários interseccionam algum objeto) e o tamanho total da imagem em *pixels*. As dimensões do quadro que o processo mestre envia para os processos escravos foram mantidas constantes em 32 x 32. Essas dimensões foram escolhidas tomando por base os resultados obtidos num trabalho que dá maior ênfase às características arquiteturais do ambiente paralelo heterogêneo [Lima et al., 1993].

5.1 Número de objetos

O gráfico da figura 1 mostra a comparação entre os comportamentos das duas técnicas em função da variação do número de objetos da cena. As imagens de teste (384 x 384 *pixels*) continham 10, 20, 30, 40 e 50 objetos. Para cada número de objetos, foram feitos testes variando a ocupação da tela de 20 a 80%. Conforme esperado, ambas as técnicas apresentaram um crescimento no tempo de execução que varia linearmente com o número de objetos, sendo a heterogênea mais eficiente em todos os casos. Também deve ser notado que a vantagem da técnica heterogênea sobre a homogênea aumenta com o número de objetos. Nas medições apresentadas, isso pode ser verificado observando o fato de que a vantagem foi de 10% para 10 objetos e aumentou progressivamente, chegando a 23% para 50 objetos.

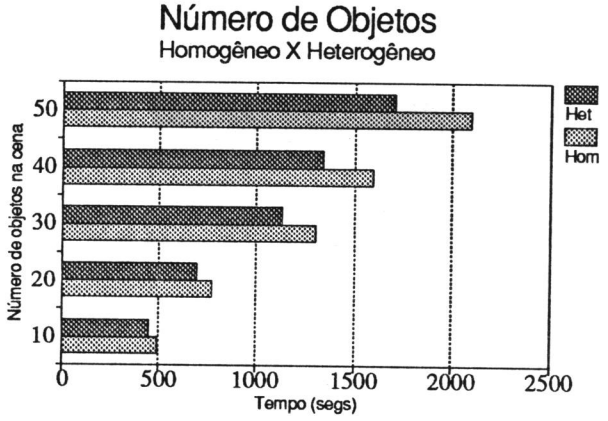

Figura 1: Comparação das técnicas homogênea e heterogênea, variando o número de objetos da cena.

5.2 Área ocupada da imagem

Na figura 2, está mostrado o gráfico comparativo das técnicas com relação à área da imagem ocupada. Foram feitos testes para 20, 50 e 80% de ocupação, gerando imagens de 384 x 384 *pixels*. Os valores mostrados no gráfico são as médias dos resultados obtidos variando o número de objetos de 10 a 50. A técnica heterogênea é superior à homogênea em todos os casos novamente. Nesse caso, ocorre um decréscimo da vantagem da técnica heterogênea sobre a homogênea quando do aumento da área da imagem ocupada. Quantitativamente, a vantagem, que foi de 17% para imagens com 20% de ocupação, caiu para 12% para imagens com 80% de ocupação. Isso ocorre porque a superioridade da técnica heterogênea vem do fato dela tratar mais eficientemente quadros degenerados e o número de tais quadros diminuir com o aumento da ocupação da tela.

5.3 Tamanho da imagem

A comparação entre os desempenhos das técnicas homogênea e heterogênea em função do tamanho da imagem está mostrada no gráfico da figura 3. Foram feitos testes para imagens de dimensões 192 x 192, 256 x 256, 320 x 320 e 384 x 384 *pixels*. Para cada tamanho de tela, foram feitas medições variando o número de objetos (de 10 a 50) e a ocupação da tela (de 20 a 80%). Os números mostrados no gráfico são a média dessas medições. Mais uma vez, a técnica heterogênea foi melhor em todos os casos. Além disso, a vantagem aumenta proporcionalmente ao tamanho da imagem. Essa vantagem é de 11% para imagens de 192 x 192 *pixels* e aumenta para 17% para imagens de 384 x 384 *pixels*. O aumento na vantagem pode ser explicado pela maior ocorrência de quadros

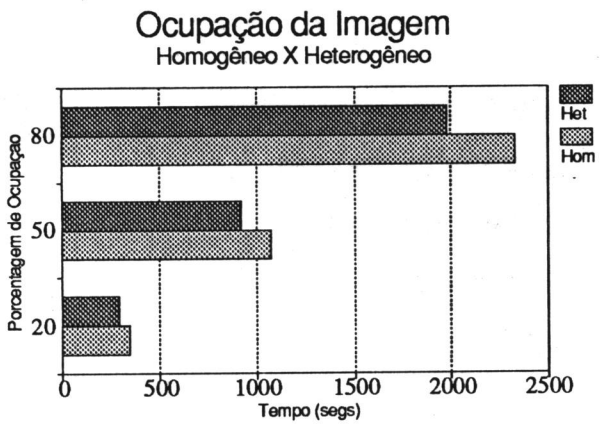

Figura 2: Comparação das técnicas homogênea e heterogênea, variando a área ocupada da imagem.

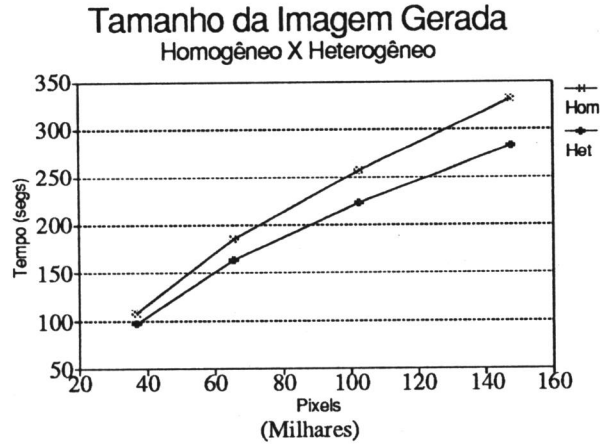

Figura 3: Comparação das técnicas homogênea e heterogênea, variando o tamanho da imagem.

degenerados em imagens maiores, uma vez que o tamanho do quadro foi mantido constante.

6 Análise da técnica heterogênea

A seguir, serão feitas algumas considerações com relação a técnica heterogênea apresentada para acelerar o processo de *ray tracing*.

Algumas técnicas de otimização obtêm seus ganhos em detrimento da liberdade de modelagem, restringindo a classe de objetos primitivos. Apesar de só ter sido implementada no contexto da geometria sólida construtiva, essa técnica pode ser aplicada a qualquer classe de objetos primitivos.

Existem técnicas de aceleração que exigem algum tipo de pré-processamento para serem aplicáveis. Em alguns casos, o uso da técnica pode até deixar de ser compensador, pois o custo de pré-processamento pode ultrapassar os benefícios trazidos por ela. A técnica heterogênea não requer nenhum tipo de pré-processamento.

Por meio da aplicação de algumas técnicas, é possível fazer com que o tempo de execução necessário para a geração de uma cena seja logarítmico ou até mesmo constante em função do número de objetos. A técnica heterogênea não apresenta ganhos em termos de ordem de complexidade, isto é, o tempo de execução ainda varia linearmente com o numero de objetos, embora seja reduzido por um fator constante.

Da mesma forma que acontece em vários métodos de aceleração, na técnica heterogênea, o ganho em termos de tempo é obtido em detrimento de um gasto adicional em memória. Para transferir o controle do processo da máquina SIMD para a MIMD, é necessário armazenar, para cada raio primário, algumas informações como sua direção, o identificador do objeto por ele interseccionado e o tempo de intersecção.

Conceitualmente, a técnica aqui apresentada é bastante simples. Entretanto, sua implementação requer que exista um ambiente paralelo heterogêneo disponível e que os algorimos sejam adaptados para se adequarem à uma arquitetura em particular. Dessa forma, perde-se uma qualidade muito desejável que é a independência com relação à máquina. Além disso, para se obter melhores resultados, é necessário determinar, empiricamente, o tamanho ótimo do quadro que é distribuído pelo processo mestre aos processos escravos na fase MIMD.

Conclusão

Este trabalho apresentou duas técnicas de aceleração de *ray tracing* por paralelismo. A primeira técnica utiliza uma máquina MIMD e a segunda utiliza um ambiente paralelo heterogêneo que utiliza uma máquina SIMD para o processamento de raios primários e uma máquina MIMD para processamento de raios secundários. Foi feita uma comparação entre essas técnicas em termos do tempo de execução gasto na geração da imagem de uma cena. A técnica heterogênea apresentou melhores resultados do que a homogênea em todos os casos estudados. Os parâmetros variados nessa comparação foram o número de objetos presentes na cena, a área ocupada da imagem e as dimensões da imagem.

Os resultados provam que a técnica de aceleração de *ray tracing* heterogênea, embora gaste mais memória e seja dependente de máquina, é superior à homogêna, pois, além de apresentar as vantagens de não restringir a classe de objetos primitivos e não exigir qualquer tipo de pré-processamento, re-

duz significativamente os tempos de excução. Como perspectiva futura, pretende-se combinar o uso da heterogeneidade com outras técnicas de aceleração, visando maiores ganhos em eficiência.

Referências

[Amanatides, 1984] Amanatides, J. (1984). Ray tracing with cones. *Comput. Graph.*, 18(3):129–135.

[Arvo and Kirk, 1987] Arvo, J. and Kirk, D. (1987). Fast ray tracing by ray classification. *Comput. Graph.*, 21(4):55–64.

[Clearly et al., 1985] Clearly, J. G., Wyvill, B. M., Birtwistle, G., and Vatti, R. (1985). Multiprocessor ray tracing. *Comput. Graph. For.*, 5:3–12.

[Cook, 1986] Cook, R. L. (1986). Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5(1).

[Dippe and Wold, 1985] Dippe, M. and Wold, E. H. (1985). Antialiasing through stochastic sampling. *Comput. Graph.*, 19(3):68–78.

[Fujimoto et al., 1986] Fujimoto, A., Tanaka, T., and Iwata, K. (1986). ARTS: Accelerated ray-tracing system. *IEEE Comput. Graph. Appl.*, 6(4):16–26.

[Geist et al., 1993] Geist, A., Benguelim, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V. (1993). *PVM 3.0 User's Guide and Reference Manual.* Oak Ridge National Laboratory.

[Gervautz, 1986] Gervautz, M. (1986). Three improvements of the ray tracing algorithm for CSG trees. *Comput. Graph.*, 10(4):333–339.

[Glassner, 1984] Glassner, A. S. (1984). Space subdivision for fast ray tracing. *IEEE Comput. Graph. Appl.*, 4(10):15–22.

[Goldsmith and Salmon, 1985] Goldsmith, J. and Salmon, J. (1985). A ray tracing system for the hypercube. Caltech Concurrent Computing Project Memorandum HM154, California Institute of Technology.

[Goldsmith and Salmon, 1987] Goldsmith, J. and Salmon, J. (1987). Automatic creation of object hierarchies for ray tracing. *IEEE Comput. Graph. Appl.*, 7(5):14–20.

[Haines and Greenberg, 1986] Haines, E. A. and Greenberg, D. P. (1986). The light buffer: a shadow testing accelerator. *IEEE Comput. Graph. Appl.*, 6(9):6–16.

[Hall and Greenberg, 1983] Hall, R. A. and Greenberg, D. P. (1983). A testbed for realistic image synthesis. *IEEE Comput. Graph. Appl.*, 3(10):10–20.

[Hanrahan, 1983] Hanrahan, P. (1983). Ray tracing algebraic surfaces. *Comput. Graph.*, 17(3):83–89.

[Heckbert and Hanrahan, 1984] Heckbert, P. S. and Hanrahan, P. (1984). Beam tracing polygonal objects. *Comput. Graph.*, 18(3):119–127.

[Jansen, 1986] Jansen, F. W. (1986). Data structures for ray tracing. In Kessener, L., Peters, F., and Lierop, M., editors, *Data Structures for Raster Graphics, Proceedings Workshop*, pages 57–73. Springer Verlag. Eurographics Seminars.

[Kajiya, 1983] Kajiya, J. T. (1983). New techniques for ray tracing procedurally defined objects. *Comput. Graph.*, 17(3):91–102.

[Kaplan, 1985] Kaplan, M. R. (1985). Space tracing a constant time ray tracer. State of the art in image synthesis. 11. Siggraph '85 Course Notes.

[Kay and Kajiya, 1986] Kay, T. L. and Kajiya, J. (1986). Ray tracing complex scenes. *Comput. Graph.*, 20(4):269–278.

[Kobayashi et al., 1987] Kobayashi, H., Nakamura, T., and Shigei, Y. (1987). Parallel processing of an object space for image synthesis using ray tracing. *The Visual Computer*, 3(1):13–22.

[Lee et al., 1985] Lee, M., Redner, A. R., and Uselton, S. P. (1985). Statistically optimized sampling for distributed ray tracing. *Comput. Graph.*, 19(3):61–67.

[Lima et al., 1993] Lima, M. A. C., Corrêa, W. T., Jr., W. M., and Carvalho, M. L. B. (1993). Uso da heterogeneidade para aceleração de *ray tracing*. Submetido para publicação.

[Max, 1981] Max, N. L. (1981). Vectorized procedural models for natural terrain: waves and islands in the sunset. *Comput. Graph.*, 15(3):317–324.

[Nemoto and Omachi, 1986] Nemoto, K. and Omachi, T. (1986). An adaptative subdivision by sliding boundary surfaces. In *Proc. of Graphics Interface '86*, pages 43–48, Vancouver, B. C.

[Nishimura et al., 1983] Nishimura, H., Ohno, H., Kawata, H., Shirakawa, T., and Omura, K. (1983). Links–1: a parallel pipelined multimicrocomputer system for imagem creation. In *Proc. of the 10th*

Symposium on Computer Architecture, pages 387–394.

[Ohta and Maekawa, 1987] Ohta, M. and Maekawa, M. (1987). Ray coherence theorem and constant time ray tracing algorithm. In Kunni, T. L., editor, *Computer Graphics 1987*, pages 303–314. Proc. of CG International '87.

[Plunkett and Bailey, 1985] Plunkett, D. J. and Bailey, M. J. (1985). The vectorization of a ray–tracing algorithm for improved execution speed. *IEEE Comput. Graph. Appl.*, 5(8):52–60.

[Purgathofer, 1986] Purgathofer, W. (1986). A statistical method for adaptative stochastic sampling. In Requicha, A. A. G., editor, *Proc. Eurographics '86*, pages 145–152, Elsevier(North–Holland).

[Roth, 1982] Roth, S. D. (1982). Ray casting for modeling solids. *Comput. Graph. Image Process.*, 18:109–144.

[Rubbin and Whitted, 1980] Rubbin, S. and Whitted, T. (1980). A three–dimensional representation for fast rendering of complex scenes. *Comput. Graph.*, 14(3):110–116.

[Sederberg and Anderson, 1984] Sederberg, T. W. and Anderson, D. C. (1984). Ray tracing of steiner patches. *Comput. Graph.*, 18(3):159–164.

[Shinya et al., 1987] Shinya, M., Takahashi, T., and Naito, S. (1987). Principles and applications of pencil tracing. *Comput. Graph.*, 21(4):45–54.

[Ullner, 1983] Ullner, M. K. (1983). *Parallel machines for computer graphics.* PhD thesis, California Institute of Technology, Pasadena, California.

[Whitted, 1980] Whitted, T. (1980). An improved illumination model for shaded display. *Commun. of CM*, 23(6):343–349.