António Augusto de Sousa

Síntese de Imagem

DEEC
FEUP
1995

António Augusto de Sousa

Síntese de Imagem

Controlo Progressivo do Nível de Realismo e Arquitecturas Paralelas

Faculdade de Engenharia da Universidade do Porto

1995

ANTÓNIO AUGUSTO DE SOUSA

Síntese de Imagem

Controlo Progressivo do Nível de Realismo e Arquitecturas Paralelas

Dissertação apresentada à Faculdade de Engenharia da Universidade do Porto para a obtenção do grau de Doutor em Engenharia Electrotécnica e de Computadores e realizada sob a orientação científica do Doutor Fernando Nunes Ferreira, Professor Catedrático da Faculdade de Engenharia da Universidade do Porto

DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES
FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO
1995

Os trabalhos de investigação apresentados nesta Tese de Doutoramento foram apoiados pelo programa PRODEP, medida 4.2, acção de formação 12.19.

AO EDUARDO
E AO DIOGO

Resumo

Neste documento, o tema base é a Síntese de Imagens por Computador. Inicia-se com um breve resumo dos principais algoritmos de síntese de imagem que são classificados em Algoritmos de Iluminação Local e Algoritmos de Iluminação Global. Seguidamente, descrevem-se os algoritmos tradicionais de Iluminação Global conhecidos por *Ray-Tracing* e Radiosidade e apresentam-se alguns dos seus problemas.

No âmbito dos Algoritmos de Iluminação Global, o algoritmo Radiosidade desempenha um papel importante. Determina a energia por unidade de tempo e de área em locais pré--seleccionados das superfícies dos objectos consideradas difusas ideais. Na fase final, a reconstrução da Função de Iluminação ao longo das superfícies é normalmente efectuada por um método de interpolação bilinear que garante a continuidade da função, mas não a continuidade das derivadas ao longo das linhas de junção. Em consequência, verifica-se o efeito de *Mach band*.

Um novo método de interpolação da Função de Iluminação é apresentado. Baseia-se na interpolação bicúbica por superfícies de Hermite. Descreve-se a forma de cálculo dos parâmetros da matriz de Hermite, nomeadamente, das derivadas parciais e mistas.

Um novo método é também apresentado para a inclusão de superfícies especulares puras em Radiosidade. A expressão de cálculo de delta factores de forma é reformulada para incluir a reflexão de energia em *patches* especulares.

O algoritmo *Ray-Tracing* é também um Algoritmo de Iluminação Global. Determina a cor de cada *pixel* da imagem de acordo com as características dos vários raios luminosos que são criados, por reflexão e transmissão. Consome grande quantidade de tempo de processamento para gerar uma imagem que é visualizada no final de todo o cálculo.

Define-se *Ray-Tracing* com Realismo Crescente e Controlável Interactivamente como uma alternativa de produzir imagens que podem ser visualizadas desde o início da sua geração e cuja qualidade, ou realismo, aumenta com o tempo de processamento. Apresentam-se duas versões experimentais para teste das novas ideias, uma das quais implementada em arquitectura paralela do tipo Multicomputador. A crítica dos resultados

desta última é feita com especial cuidado, tendo em vista a descrição das potencialidades de uma versão melhorada, usando o mesmo tipo de arquitectura.

Descreve-se o sistema IIRRA-*Interactive Increasing Realism Ray Tracing Algorithm*. Definem-se os seus módulos constituintes na base de um sistema de *Ray-Tracing* com Nível Crescente de Realismo, implementado sobre uma arquitectura multicomputador com memória virtual partilhada, para optimizar a quantidade de memória disponível no sistema.

Descreve-se a hierarquia de acessos a memória partilhada, baseada em *caches* locais e definem-se métodos de optimização de acessos a dados remotos com incidência na redução do tráfego de mensagens na rede de processadores. Discute-se o balanceamento da carga computacional.

Define-se um novo tipo de endereçamento *multicast*, designado por *InPathTo*, criado para facilitar os acessos a dados e a tarefas localizados remotamente. Efectua-se um estudo analítico da utilização do mesmo, com vista à determinação das melhores condições de eficiência.

Avaliam-se dois métodos de *Routing* com capacidade para evitar o *Deadlock*.

Abstract

The main subject of this document is Image Synthesis by Computer. It begins with a short summary of the most important algorithms used in image synthesis and their classification in Local Illumination Algorithms and Global Illumination Algorithms. Next, the traditional algorithms for computing Global Illumination known as Ray-Tracing and Radiosity and some of their problems are presented.

The Radiosity algorithm performs a very important role in the field of Global Illumination. It computes the amount of energy per time and area units, in pre-selected places of the objects surfaces. Every surface is considered ideal and strictly diffuse. In the final phase, it performs the reconstruction of the Illumination Function for all the surfaces, using bilinear interpolation. This method allows the continuity of the illumination function, but not the continuity of the illumination derivative, along the junction lines, presenting the Mach Band effect.

A new method for the interpolation of the Illumination Function is presented. It is based on the bicubic interpolation using Hermite surfaces. The evaluation of the partial derivatives, and other parameters of the Hermite matrix are presented.

It is also presented a new method for the inclusion of perfect specular surfaces in Radiosity. The mathematical expression of the delta form factors is rearranged, in order to include the reflected energy in specular patches.

The Ray-Tracing is also a Global Illumination algorithm. It computes the colour of each image pixel based on the characteristics of several light rays that are created by reflection and by transmission. The Ray-Tracing algorithm needs a large amount of processing time to generate one single image. The visualisation of the computed image is only possible after the end of processing.

Increasing Realism and Interactively Controlled Ray-Tracing can be defined as an alternative way to generate images that can be visualised from the beginning. The quality or realism of the image increases with the processing time. Two experimental versions are presented. One of them is implemented in a Multicomputer parallel architecture. The

discussion of the results from this implementation is done with a special care, having in mind a description of an improved version to be implemented on a similar architecture.

The IIRRA-Interactive Increasing Realism Ray Tracing Algorithm system is presented. All its modules are described on a basis of a Increasing Realism Ray-Tracing System, implemented on a multicomputer architecture with shared virtual memory, to optimise the total amount of memory.

The hierarchy of shared memory accesses based in local caches, is presented, together with some optimisation methods of remote data accesses, developed in order to reduce the message traffic on the network of processors. A discussion of the load balancing is also presented.

A new kind of multicast addressing type, called InPathTo, is defined. It has been developed to allow an easier access to remote data and tasks. To select the best efficiency conditions, a mathematical analysis of this addressing behaviour is done.

Résumé

Dans ce document, le thème de base est la Synthèse d'Images para Ordinateur. Au début est présenté un bref résumé des principaux algoritmes de synthèse d'image qui sont classifiés d'Algoritmes d'Illumination Locale e d'Algoritmes d'Illumination Globale. Ensuite, sont décrits les algoritmes traditionnels d'Illumination Globale connus par *Ray-Tracing* et Radiosité et sont présentés quelques uns de leurs probèmes.

Dans l'environnement des Algoritmes d'Illumination Globale, l'algoritme Radiosité joue un rôle important. Il détermine l'énergie par unité de temps et d'aire en des endroits préselectionnés des surfaces des objects considérées difuses idéales. Dans la phase finale, la reconstruction de la Fonction d'Illumination au long des surfaces est normalement effectuée par une métode d'interpolation bilinéaire qui garantie la continuité de la fonction, mais pas la continuité des dérivées au long des lignes de jonction. Par conséquent, on vérifie l'effet de *Match Band*.

Une nouvelle métode d'interpolation de la Fonction d'Illumination est présentée. Elle est basée sur l'interpolation bicubique par surfaces d'Hermite. On décrit la forme de calcul des paramètres de la matrice d'Hermite, notamment, des dérivées partielles et mixtes.

Une nouvelle métode est présentée aussi par l'inclusion de surfaces spéculaires pures em Radiosité. L'expression de calcul de delta facteurs de forme est reformulée pour incluir la réflection d'énergie em *patches* spéculaires.

L'algoritme *Ray-Tracing* est aussi um Algoritme d'Illumination Globale. Il détermine la couleur de chaque pixel de l'image d'accord avec les caractéristiques des divers rayons lumineux qui sont créés, par réflection et transmission. Il consomme une grande quantité de temps de processus pour gérer une image qui est visualisée en fim de chaque calcul.

On définit *Ray-Tracing* à Réalisme Croissant et Controlable Interactivement comme une alternative de produire des images qui peuvent être visualisées depuis le début de leur gestion e dont la qualité, ou réalisme, augmente avec le temps de processus. Deux versions experimentales sont présentées pour le test des nouvelles idées, une des quelles implementée en architecture paralelle du type Multiordinateur. La critique des résultats

de celle-ci est faite avec une spéciale attention, visant la description des potentialités d'une version améliorée, utilisant le même type d'architecture.

On décrit le système IIRRA-*Interactive Realisme Ray Tracing Algorithm*. On définit ses deux modules constituants se basant sur un système de *Ray-Tracing* à Niveau Croissant de Réalisme, implémenté sur une architecture multiordinateur à mémoire virtuelle partagée, pour optimiser la quantité de mémoire disponible dans le système.

On décrit la hiérarchie d'accés à la mémoire partagée, basée sur *caches* locales e on définit des métodes d'optimisation d'accés à des données à distance avec incidence sur la réduction du trafique de messages sur le réseau de processeurs. On discute le balancement de la charge computationelle.

On définit un nouveau type d'adressage *multicast*, désigné par *InPathTo*, créé pour faciliter les accés à données et à tâches à distance. On effectue une étude analitique de son utilisation, visant à déterminer les meilleures conditions d'efficacité.

On évalue deux métodes de *Routing* capables d'éviter le *Deadlock*.

Agradecimentos

O desenrolar dos trabalhos conducentes a um doutoramento e, na fase final, a escrita da respectiva tese são tarefas complexas, demoradas e, muitas vezes, desesperantes. Nos momentos de maior desespero, um conselho técnico, uma conversa amiga ou simplesmente uma paciente escuta, são preciosos alentos.

Não é possível mencionar aqui todas as pessoas que, de uma forma ou de outra, me deram um qualquer desses apoios. Fica o meu muito obrigado.

Das pessoas que mais directamente contactaram comigo durante estes anos, faço questão de deixar marcados, nestas pobres páginas e de uma forma indelével, os seus nomes e expressar-lhes os meus mais profundos agradecimentos.

- Em primeiro lugar, a pessoa de quem tenho um orgulho muito grande em dizer que fui seu orientando, o Professor Fernando Nunes Ferreira. Pelos sábios conselhos que deu, pelo amigo que foi e por tudo quanto sofreu em paralelo comigo, como se de um trabalho seu se tratasse. Não há palavras que cheguem...

- O Vasco Branco. Colega e amigo de (já longos) anos de caminhada conjunta em direcção a esta meta tão desejada. Obrigado Vasco e conta comigo.

- O António Costa, o Miguel Leitão e o Rui Bastos. Pelos trabalhos que realizaram e os quais tenho o maior orgulho em referir nesta Tese. Pela boa vontade com que atenderam sempre aos meus pedidos. Também pelas discussões técnicas e pela paciência que tiveram para comentarem alguns dos capítulos que a compõem.

- Os alunos que comigo trabalharam em projectos de fim de curso e, muito especialmente, o Hugo Ferreira, pelo seu entusiasmo e pelos fins de semana que perdeu, depois de ter finalizado o curso, para levarmos a termo um dos trabalhos que aqui é apresentado.

- O Grupo de Computação Gráfica & CAD do INESC em geral, por todas as ajudas prestadas. Não menciono pessoas.

Desejo agradecer às instituições envolvidas, FEUP e, especialmente, INESC, os meios que me disponibilizaram, logísticos e de equipamento.

Também quero pedir desculpas. À minha família. Foram longas horas estas, em que o convívio familiar foi ultrapassado por "ter que ser" ainda mais fortes. Prometo que vou passar a ser mais bem disposto... e que vou deixar de fumar...

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 0

Introdução

0. Introdução

Vive-se hoje sob o domínio da imagem, um meio privilegiado para transmitir qualquer tipo de informação.

No final de século em que a revolução informática sucedeu à revolução industrial, o computador é um agente fundamental na produção de imagens. A sua aplicação estende-se desde a concepção de objectos até à geração de sequências de imagens que dão corpo a muitos dos filmes publicitários que se consomem diariamente através da televisão.

0.1 Motivação

A Síntese de Imagens é uma área da Computação Gráfica dedicada à geração de imagens de cenas tridimensionais a partir de um modelo matemático adequado.

A algoritmia de síntese de imagem varia de acordo com as aplicações. Algoritmos mais simples e rápidos com resultados qualitativamente mais baixos, para tarefas interactivas de visualização de objectos em sistemas de CAD (*Computer Aided Design*) e, no extremo oposto, algoritmos mais lentos mas capazes de produzir imagens com qualidade quase fotográfica.

Os primeiros algoritmos de síntese de imagens datam da década de setenta. Nessa altura, o fraco desempenho dos computadores era um óbice ao desenvolvimento de algoritmos sofisticados e os requisitos em termos de imagens de síntese eram baixos.

Uma parte desses algoritmos pioneiros da síntese de imagens são, ainda hoje, utilizados, nomeadamente no apoio à visualização em sistemas de CAD. A maior parte deles baseia-se em algoritmos de visibilidade com cálculo de iluminação local ou seja, iluminação directa pelas fontes de luz.

No entanto, os requisitos actuais são outros, assim como as máquinas disponíveis. Por isso, os algoritmos de iluminação global são uma presença constante na produção imagética associada ao quotidiano mediático actual.

Porém, os problemas da síntese de imagem estão ainda longe de completamente resolvidos. É constante a procura de mais e melhores algoritmos cuja complexidade vai acompanhando, ou mesmo ultrapassando, o incremento do desempenho das máquinas. Se, nos primeiros tempos, uma imagem de uma esfera aproximada por polígonos preenchidos com iluminação constante era considerada razoável, hoje espera-se que um algoritmo de iluminação global seja capaz de reproduzir fielmente as várias interacções luminosas entre objectos.

Compreende-se desse modo que os algoritmos de síntese de imagem actuais continuem a consumir largas horas de processamento para criarem uma imagem de uma cena complexa. Grande número dos trabalhos publicados na área da síntese de imagem dedica-se ao objectivo final de acelerar tanto quanto possível os algoritmos respectivos.

Outras visões do problema podem no entanto ser enunciadas.

Na fase inicial de um projecto, quer seja na área do *design* industrial ou no domínio do *design* de comunicação, um utilizador pode fazer uso da síntese computacional de imagem para experimentar as várias soluções que tem em mente, de modo a filtrar as que lhe são mais convenientes. Nesta fase do trabalho, não é muito exigente com o rigor das imagens que obtém, bastando-lhe a obtenção de algo semelhante a esboços.

Nas fases finais do projecto, o número de soluções que pretende testar é bastante inferior, mas o rigor que pretende das imagens é muito maior.

No fundo, o mesmo utilizador, no mesmo projecto, tem carências diferentes em termos de qualidade das imagens geradas.

Uma solução possível para este tipo de problema passa pela disponibilização de dois ou mais métodos de síntese de imagem. Um com baixa qualidade de imagem, mas bastante

rápido, para as primeiras fases do projecto, outro com grande qualidade de imagem, mas mais penalizado em termos de tempo de processamento e sendo de considerar, eventualmente, outros métodos intermédios. Tal solução não parece ser muito adequada dado que, por um lado, a coexistência de vários métodos encarece os sistemas e, por outro lado, a mudança brusca de nível de realismo da imagem conduz, provavelmente, a situações inesperadas.

Uma potencial solução passa por redefinir um algoritmo de iluminação global, de modo que seja possível produzir imagens com diferentes níveis de realismo, de uma forma progressiva, isto é, baixo realismo, adequado às primeiras fases do projecto e, com o decurso de mais tempo de processamento, elevado realismo que satisfaça as necessidades do utilizador nas fazes finais do projecto. Parece apropriado designar um tal algoritmo por **Síntese de Imagens com Realismo Progressivo**.

Lucros adicionais podem ser retirados dessa estratégia. Mesmo nas fases finais de um projecto, sucede muitas vezes que o utilizador inicia a geração de uma imagem e, bastante mais tarde, conclui que não corresponde à imagem que tinha idealizado. Ao acompanhar a progressão da qualidade da imagem, o utilizador pode verificar bastante mais cedo que, eventualmente, a imagem que está a obter não corresponde à imagem mental que criou acerca daqueles objectos e, nesse caso, pode interromper o processo.

Mais ainda, a forma como o realismo da imagem progride no tempo pode não ser única. Neste caso, decisões tomadas pelo utilizador podem ser de extrema utilidade, se lhe for permitido controlar, por algum meio, o realismo da imagem que está a gerar. É apropriado designar um tal processo por Síntese de Imagens com **Controlo Progressivo do Nível de Realismo**.

O realismo de uma imagem depende de vários factores. Em função do algoritmo utilizado, a imagem pode ser mais realista em alguns aspectos, mas menos realista noutros. Por exemplo, existem neste momento, dois tipos de algoritmos de iluminação global, que induzem a imagens bastante realistas, o *Ray-Tracing* e a Radiosidade (*Radiosity*), que exibem características com uma certa complementaridade.

O algoritmo *Ray-Tracing*, inicialmente introduzido por T. Whitted [Whi80], exibe um realismo acentuado na simulação dos efeitos ópticos especulares e falha, de certo modo, na simulação dos efeitos ópticos difusos. O algoritmo radiosidade, introduzido na área da síntese de imagem por C.Goral *et al.* [Gor84], admite somente superfícies puramente difusas. A complementaridade referida é, neste caso, óbvia e interessa ser explorada.

Além das limitações mencionadas, os algoritmos em causa apresentam várias outras, algumas das quais serão tratadas no decorrer deste documento, nomeadamente, o tempo

excessivo de processamento associado ao cálculo de uma imagem em *Ray-Tracing* e a reconstrução da Função de Iluminação em Radiosidade.

Os sistemas de arquitectura paralela são uma via interessante para a aceleração de algoritmos e constituem, por si só, um tópico de investigação aliciante.

A disponibilização de arquitecturas desse tipo por volta dos anos 80, baratas, modulares e expansíveis, aumentou o interesse pela sua utilização nos sistemas de síntese de imagem. O algoritmo *Ray-Tracing*, apresenta uma vocação natural para a paralelização e, neste contexto, vários trabalhos de investigação têm sido publicados.

A paralelização do algoritmo *Ray-Tracing* constitui uma forma de acelerar o seu processamento, o que gera um ambiente favorável ao desenvolvimento da ideia de progressão de realismo mencionada anteriormente.

O projecto **IIRRA** (*Increasing Realism Ray-tracing Algorithm*) surge na confluência das direcções de investigação referidas, ou seja, **Realismo Crescente Controlável de um modo Interactivo**, com base em **Arquitecturas Paralelas**.

0.2 Organização do Documento

A organização deste documento traduz um conjunto de trabalhos desenvolvidos no domínio da Síntese de Imagens pelo autor, ou com a sua participação.

O documento encontra-se organizado em seis capítulos que espelham uma preocupação não só de carácter científico, mas também pedagógico, oferecendo ao leitor o corpo teórico julgado suficiente para a compreensão do trabalhos apresentados.

No **primeiro capítulo**, dividido em duas partes, apresenta-se um resumo de assuntos julgados relevantes para a síntese de imagens. A primeira dedica-se à classificação dos algoritmos de iluminação local e descrição de alguns julgados mais marcantes. Na segunda parte, apresentam-se os dois algoritmos de iluminação global, *ray-tracing* e radiosidade que serão utilizados nos desenvolvimentos apresentados nos capítulos 3, 4 e 5.

Alguns desses desenvolvimentos, dedicados ao algoritmo *ray-tracing*, utilizarão como base uma arquitectura paralela. Justifica-se desta forma a síntese que é efectuada, no **segundo capítulo**, de temas que se relacionam com os sistemas de arquitectura paralela em geral e, em particular, com implementações do algoritmo *ray-tracing*. Neste capítulo, efectua-se também uma apresentação dos sistemas baseados em *Transputers*, assim como

um resumo da sintaxe e das construções da linguagem de programação OCCAM, utilizados nos capítulos seguintes.

O **terceiro capítulo** é dedicado à descrição de dois novos desenvolvimentos efectuados sobre o algoritmo radiosidade, um na reconstrução de funções de iluminação e outro na inclusão de superfícies especulares.

O **quarto capítulo** define a noção de realismo crescente em *ray-tracing* e descreve os estudos efectuados, nesse contexto, sobre uma versão sequencial e outra em arquitectura paralela baseada em *transputers*. Os resultados apresentados servem de base à definição de uma versão melhorada de *ray-tracing* com realismo crescente sobre uma arquitectura paralela, a apresentar no quinto capítulo.

No desenvolvimento dessa versão melhorada, foi necessário recorrer a técnicas especiais de paralelização, afim de optimizar alguns aspectos. No **quinto capítulo**, descrevem-se essas técnicas, algumas das quais apresentam alguma originalidade, assim como as formas adoptadas para melhorar a implementação do crescimento do realismo.

Finalmente, no **sexto capítulo** resumem-se as principais conclusões extraídas ao longo do documento e apontam-se as pistas de futuros trabalhos nas mesmas áreas.

Capítulo 1

Síntese de Imagens

1. Síntese de Imagens

A Síntese de Imagens consiste na criação de imagens a partir de um modelo geométrico e de um modelo de propagação de luz. É uma componente importante da Computação Gráfica que tem sido extensivamente estudada desde a década de 70.

A capacidade de criar imagens de ambientes ou cenas tridimensionais não existentes é importante para aplicações que vão desde o *design* industrial e arquitectura até à publicidade e entretenimento. Também se torna um meio fundamental, na área da Visualização Científica, para representar dados de fenómenos normalmente não acessíveis à vista (por exemplo, os obtidos por ressonância magnética). A síntese de imagens toma, neste caso, o nome de Visualização de Volumes (*volume rendering*).

Os objectivos da Síntese de Imagens têm evoluído ao longo dos tempos, muito principalmente devido às limitações tecnológicas dos computadores. Na década de 70, os objectivos eram somente a representação aproximada dos objectos, sem grandes preocupações de aplicação de texturas, projecção de sombras, etc. Hoje em dia, busca-se a simulação perfeita das interacções da luz nos vários objectos, utilizando-se modelos de propagação e reflexão da luz o mais próximo possível das Leis da Física, no sentido de obter uma imagem tão perfeita quanto uma fotografia.

A aplicabilidade de imagens de elevado nível de realismo está no entanto condicionada pelo actual estado da arte, quer ao nível das capacidades de processamento das máquinas quer mesmo ao nível dos actuais algoritmos de síntese de imagens. Torna-se assim

necessária uma análise da relação Custo Computacional/Nível de Realismo (que passa a ser designada somente por relação custo/realismo) dos vários algoritmos elegíveis para uma dada aplicação.

Na extremidade mais baixa da relação custo/realismo temos os algoritmos de desenho com representação por linhas (tradução normalizada do termo inglês *wire-frame*). Embora ofereçam um nível de realismo extremamente baixo, exigem um custo computacional desprezável, pelo que permitem facilmente (mesmo com máquinas de baixo desempenho) obter, por parte dos sistemas, resposta em tempo real. Alguns melhoramentos podem inclusivamente ser introduzidos no sentido de melhorar o realismo das imagens, nomeadamente por recurso a técnicas de *depth-cueing*[1].

Na outra extremidade, temos os algoritmos ditos de Iluminação Global por determinarem, em cada ponto da superfície de um objecto, a participação da iluminação de outros pontos da cena. As imagens criadas por estes algoritmos podem ser quase tão realistas como uma fotografia, mas com um custo computacional enorme (uma imagem complexa pode exigir várias horas ou mesmo dias de tempo de cálculo).

Entre as duas extremidades encontram-se os algoritmos que, por oposição aos últimos, se designam por algoritmos de Iluminação Local. Nestes, a iluminação de um ponto deve-se unicamente à iluminação directa desse ponto por uma fonte luminosa em cena. Fundamentalmente, estes algoritmos determinam a cor de um ponto com base nas características de brilho da fonte de luz e na posição desta relativamente ao ponto a iluminar, entrando em conta, em alguns casos, com o facto de os dois pontos serem ou não mutuamente visíveis. Considera-se que a relação custo/realismo destes algoritmos é baixa/média.

Neste capítulo descreve-se o problema da Iluminação Local e, neste contexto, apresentam-se alguns algoritmos de visibilidade, considerados mais importantes. Define-se Iluminação Global e apresentam-se dois algoritmos, *Ray-tracing* e Radiosidade (*Radiosity*), considerados como sendo de Iluminação Global. A importância que estes dois últimos têm, para o desenvolvimento dos trabalhos conducentes à elaboração deste documento, justifica que sejam alvo de uma atenção especial.

[1] *Depth-cueing*: diminuição, com a distância ao observador, da intensidade do traço ao longo de um segmento de recta.

1.1 Algoritmos de Iluminação Local

Nestes algoritmos, o cálculo da iluminação de um ponto efectua-se com base na posição relativa entre fonte de iluminação, o próprio ponto e o observador. Parte destes modelos são empíricos, muito embora alguns apresentem alguns aspectos que se baseiam nas Leis da Física.

Um modelo de iluminação deste tipo é o clássico Modelo de Reflexão de Phong [Pho75]. Embora empírico, foi bastante utilizado por exibir uma qualidade de imagem razoável a um custo computacional baixo e encontra-se amplamente descrito na bibliografia da especialidade [Fol90], [Wat92].

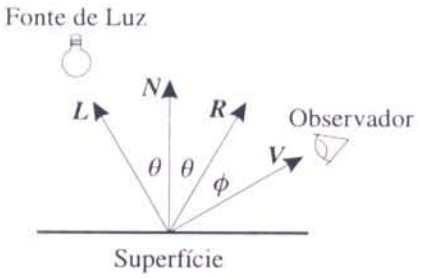


Fig. 1.1 - Geometria do modelo de reflexão de Phong

A equação do modelo de Phong relaciona a Intensidade de Luz reflectida numa superfície de um objecto e vista por um observador com as características geométricas da reflexão (Fig. 1.1):

$$I_{\lambda,r}(\lambda,\phi) = I_{\lambda,a} k_a(\lambda) + I_{\lambda,i}\left[k_d(\lambda)\cdot(L\cdot N) + W(\theta)\cdot(R\cdot V)^n\right] \qquad \text{Eq. 1.1}$$

Com:

- $I_{\lambda,r}$ intensidade reflectida no comprimento de onda λ segundo a direcção de reflexão ϕ;
- $I_{\lambda,a}$ termo que modela a luz ambiente (considerada como constante e independente da direcção);
- $I_{\lambda,i}$ intensidade de luz incidente na superfície, com origem na fonte de luz;
- k_a coeficiente de reflexão da superfície, relacionado com a componente de iluminação ambiente;
- K_d coeficiente de reflexão relacionado com a componente difusa da iluminação (independente da direcção de visão); é também conhecido como sendo a fracção de energia reflectida de modo difuso;

$W(\theta)$	fracção de energia reflectida de modo especular; embora seja função do ângulo de reflexão, aproxima-se vulgarmente por uma constante K_s;
n	valor que controla a pureza da reflexão especular (quanto maior n, mais a superfície se aproxima do espelho puro);
N	vector normal à superfície, normalizado;
L, R, V	vectores normalizados orientados respectivamente para a fonte de luz, para a direcção de reflexão óptima e para o observador;
$(L \cdot N)$	cosseno do ângulo de incidência θ;
$(R \cdot V)$	cosseno do ângulo de observação ϕ.

Nos algoritmos que utilizam o paradigma da iluminação local, o principal problema que se coloca é a determinação da visibilidade entre dois pontos da cena. Em primeiro lugar, para eliminar da imagem as partes dos objectos que não são visíveis, devido à opacidade de objectos colocados entre eles e o observador (visibilidade calculada entre os objectos e ponto de observação). Em segundo lugar, para iluminar os pontos dos objectos que são visíveis por essa fonte de luz (aplicando-se neste caso um modelo de reflexão local, por exemplo o de Phong), operação que é vulgarmente conhecida por projecção de sombras e que corresponde a trocar o observador por uma fonte de luz.

São conhecidos vários algoritmos de visibilidade (*hidden surface* ou *hidden line algorithms*), a maioria dos quais data dos anos 70 e admitem objectos constituídos por faces planas. I. Sutherland *et al.*, em [Sut74], apresentam dez algoritmos de visibilidade e estabelecem a seguinte taxonomia:

Algoritmos no Espaço Objecto
- à aresta
- ao volume

Algoritmos do tipo Lista de Prioridade
- *à priori*
- lista dinâmica

Algoritmos no Espaço Imagem
- à área
- ao ponto (ou à linha de varrimento)

Um algoritmo do tipo Espaço Objecto concentra-se na relação geométrica entre objectos no modelo de descrição da cena, com o fim de determinar que partes desses objectos são visíveis. Pelo contrário, um algoritmo no Espaço Imagem determina, para cada ponto da imagem, qual o objecto que é visível. Os algoritmos no Espaço Objecto trabalham com uma precisão arbitrária, podendo a imagem final ser aumentada várias vezes sem perda

de qualidade, enquanto que os algoritmos no Espaço Imagem se vêem limitados a uma precisão bastante inferior, vulgarmente a precisão do próprio dispositivo de visualização.

Os algoritmos do tipo Lista de Prioridade ficam situados entre os dois casos anteriores, dado que funcionam parcialmente em cada espaço. Os cálculos de profundidade são efectuados com precisão elevada, enquanto que as imagens são calculadas com a resolução disponível no sistema de visualização.

Do conjunto de algoritmos de visibilidade apresentados por I. Sutherland *et al.*, salientam-se, pela ampla utilização que têm tido integrados em sistemas de Desenho Assistido por Computador (*CAD Systems*), os do tipo Lista de Prioridade Dinâmica e do tipo Imagem à Linha de Varrimento.

1.1.1 Algoritmo de Newell *et al.*

O algoritmo de Newell *et al.* [New72], é classificado por I. Sutherland *et al.* como um algoritmo do tipo Lista de Prioridade Dinâmica. Baseia-se na construção de uma lista de faces, ordenada segundo um critério de prioridade: uma face P tem prioridade sobre outra Q, se P oculta total ou parcialmente Q.

O algoritmo divide-se em duas fases principais:

1. Pré-ordenação de todas as faces por ordem crescente da profundidade (coordenada Z) do vértice mais afastado do ponto de observação (figura 1.2);
2. Cálculo de prioridades finais entre faces.

Não sendo detectada, durante a primeira fase, qualquer sobreposição em profundidade entre faces, então a ordem resultante corresponde à ordem de prioridades finais desejada. Caso contrário, um algoritmo mais complexo decide, na segunda fase, quais as faces que têm prioridade superior, recorrendo, quando necessário, a operações de corte de uma face em outras menores.

Uma vez definida a prioridade entre faces, estas são enviadas para o dispositivo de visualização, começando pelas de menor prioridade.

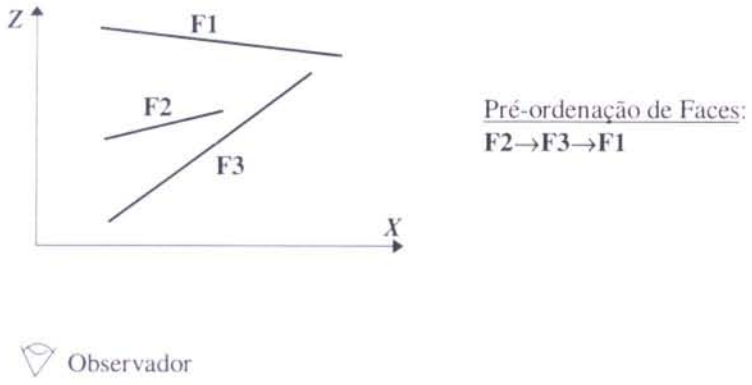


Fig. 1.2 - Pré-ordenação de faces, no algoritmo de Newell *et al.*, pela profundidade *Z* dos vértices mais afastados do observador

1.1.2 Algoritmos de Watkins

Os algoritmos de Romney [Rom70], de Bouknight [Bou69] e de Watkins [Wat70] (conhecidos vulgarmente pela família de algoritmos de Watkins) são referidos por I. Sutherland *et al.* como sendo algoritmos de visibilidade no espaço imagem, ao ponto ou linha de varrimento. Discretizam o ecrã em linhas de varrimento, fazendo corresponder uma linha de varrimento a um plano horizontal *ZX* como se mostra na figura 1.3(a). A intersecção deste plano com as faces dos objectos definem segmentos de recta (figura 1.3(b)) que, por serem entidades 2D, tornam mais fácil a atribuição de prioridades do que as faces originais, 3D.

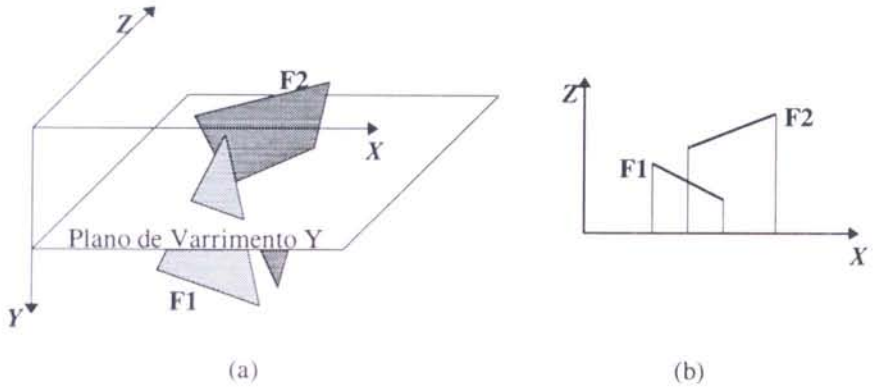


Fig. 1.3 - Discretização do espaço 3D por meio de planos de varrimento

Esta família de algoritmos explora fortemente a noção de coerência da imagem, quer vertical (semelhança entre uma linha de varrimento e a anterior), quer horizontal

(grandes porções de uma linha de varrimento mantêm a mesma prioridade de segmentos de faces) e efectua a eliminação de faces ocultas.

Numa fase preliminar do desenrolar dos trabalhos conducentes à elaboração do presente documento, desenvolveu-se um algoritmo de visibilidade que, baseando-se nos princípios básicos da família de algoritmos de Watkins e nos trabalhos apresentados em [Bou80], permite também a eliminação de arestas ocultas [Sou87], [Sou88]. Ao efectuar a análise de uma linha de varrimento, o algoritmo percorre a lista de arestas activas (AEL) dessa linha e determina quais os segmentos de faces visíveis. Se os segmentos de faces visíveis à esquerda e à direita de uma aresta são os mesmos, a aresta é invisível. Caso contrário, a aresta é visível. Quando, numa linha de varrimento, a aresta muda de estado visível para invisível, é desenhada a partir do ponto em que se tornou visível.

1.1.3 Algoritmo de Atherton & Weiler

O algoritmo de Atherton & Weiler é posterior à publicação referida de I. Sutherland *et al.* Classifica-se no grupo dos algoritmos no espaço objecto.

Embora o cálculo de iluminação com projecção de sombras seja uma operação idêntica ao cálculo de visibilidade (observador substituído pela fonte de luz), os algoritmos mencionados não oferecem muitas facilidades para a sua realização: a família de algoritmos de Watkins produz resultados em coordenadas imagem, sem qualquer relação com os objectos originais e o algoritmo de Newell *et al.* efectua uma ordenação dos polígonos por ordem crescente de distância ao observador (ou fonte de luz) sem os classificar como visíveis (iluminados) ou invisíveis (na sombra).

O algoritmo de Atherton & Weiler [Wei77] é uma solução possível para o problema mencionado. Como dados de entrada recebe a lista de polígonos em cena e produz duas listas, para os polígonos visíveis e invisíveis,[1] como se pode ver na figura 1.4. O formato dos dados de saída é idêntico ao formato dos dados de entrada, o que permite correr o algoritmo duas ou mais vezes consecutivas, colocando as fontes de luz na posição do observador para determinar os polígonos em sombra. Uma última corrida com o observador em posição gera a informação necessária à imagem final, podendo então aplicar-se um modelo de iluminação local.

[1] Os polígonos de saída resultam do recorte dos polígonos de entrada de acordo com as porções visíveis e invisíveis destes.

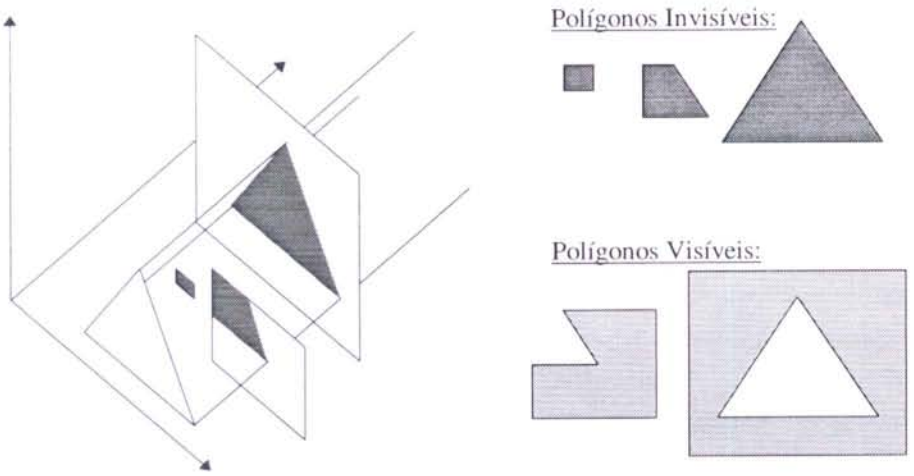


Fig. 1.4 - Recorte de polígonos no algoritmo de Atherton & Weiler

1.1.4 Algoritmo *Z-buffer*

Este algoritmo classifica-se, na taxonomia de I. Sutherland *et al.* no grupo dos algoritmos no espaço imagem.

O algoritmo *z-buffer*, desenvolvido por Catmull [Cat74], tornou-se um dos mais utilizados algoritmos de visibilidade. Este facto deve-se a uma grande simplicidade que lhe permite uma fácil implementação quer em *software* quer em *hardware*.

O algoritmo necessita de duas áreas de memória, uma para construção da imagem (designada por *frame buffer*) e outra para armazenamento de profundidades Z (designada por *z-buffer*). Efectua o varrimento linha a linha dos polígonos, um de cada vez, numa ordem arbitrária. Se, num dado ponto, a profundidade Z de um polígono é inferior à profundidade armazenada anteriormente nas mesmas coordenadas, então o polígono é visível nesse ponto, e o algoritmo substitui, nas duas áreas de memória, os valores anteriores pelos valores calculados para o novo polígono, naquele ponto.

Alguns desenvolvimentos mais recentes do algoritmo contemplam a redução do efeito de escada[1] (*A-Buffer*, [Car84]) e a projecção de sombras (*z-buffer* em dois passos, [Wil78]).

[1] Efeito de Escada é a tradução normalizada do termo inglês *aliasing*.

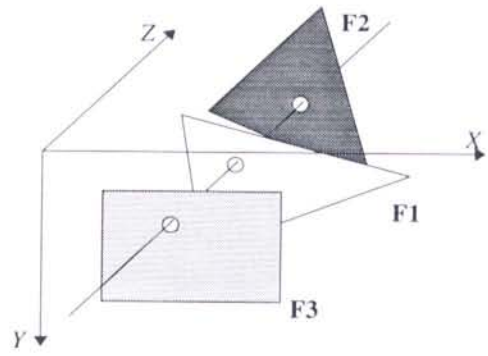


Fig. 1.5 - Princípio de funcionamento do algoritmo *z-buffer*

1.1.5 Cross Scanline

O algoritmo *Cross Scanline* insere-se no grupo dos algoritmos no espaço imagem, segundo a taxonomia de I. Sutherland *et al.*

A redução do efeito de escada tem sido sede de muitos estudos no âmbito dos algoritmos de visibilidade. É possível efectuar essa redução determinando a cor de cada *pixel* pela média das cores dos polígonos que o intersectam. A média é pesada pela área que cada polígono ocupa dentro do *pixel*.

A determinação da área de um polígono circunscrita a um *pixel* pode ser efectuada por uma operação de recorte do polígono pela janela rectangular que corresponde ao *pixel*. É um método eficaz, mas fortemente consumidor de tempo de cálculo.

No caso particular dos algoritmos à linha de varrimento, aquela área pode ser calculada por meio da definição de linhas de varrimento intermédias a que se chamam sublinhas. As sublinhas podem ser definidas de uma forma regular (discretização em um número de linhas superior ao do ecrã) ou de uma forma adaptativa (as sublinhas são definidas de forma a passarem por pontos particulares como sejam os vértices e as intersecções das arestas). As sublinhas cortam os polígonos em trapézios ou triângulos cuja área é fácil de determinar, mas as áreas são sempre calculadas de uma forma aproximada.

Em [Tan90], apresenta-se um algoritmo designado por *Cross Scanline* que calcula com exactidão as áreas dos polígonos circunscritos a um *pixel* e em [Tan94], mostra-se como o algoritmo pode, à custa de grandes quantidades de memória, ser utilizado para produzir imagens às quais se podem aplicar mudanças de escala sem perda de qualidade, mapear texturas e determinar sombras projectadas. A base do algoritmo é um varrimento simples

na direcção horizontal seguido de um varrimento adaptativo vertical. Uma linha de varrimento vertical é traçada entre um par de linhas horizontais consecutivas, passando pelos pontos de intersecção da linha horizontal com as arestas, pelos vértices dos polígonos ou pelos pontos de intersecção de arestas.

1.2 Algoritmos de Iluminação Global

Os algoritmos de iluminação global encontram-se na extremidade superior da relação custo/realismo dos algoritmos de visualização por permitirem a obtenção de imagens que produzem um estímulo visual semelhante ao que se obtém com reproduções fotográficas.

A designação Algoritmos de Iluminação Global deve-se ao facto de a iluminação de um ponto de um objecto depender não só da luz que lhe chega directamente das fontes de luz mas também da luz que lhe chega após reflexão e/ou transmissão de luz nas superfícies de outros objectos.

A Equação de *Rendering* [Kaj86] traduz matematicamente esta noção e apresenta-se de seguida. Repare-se na natureza recursiva da equação, traduzida pela grandeza I e ainda no integral, aplicado ao domínio S, que contém todos os pontos x.

$$I(x,x') = g(x,x') \cdot \left[\varepsilon(x,x') + \int_S \rho(x,x',x'') \cdot I(x',x'') \cdot dx'' \right] \qquad \text{Eq. 1.2}$$

com:

$I(x,x')$	Intensidade de luz que passa do ponto x' para o ponto x
$g(x,x')$	Termo geométrico caracterizado pela visibilidade mútua e pela distância entre x e x'
$\varepsilon(x,x')$	Intensidade de luz emitida do ponto x' para o ponto x
$\rho(x,x',x'')$	Intensidade de luz que, partindo do ponto x'', atinge o ponto x após reflexão no ponto x'.

Em [Fol90], classificam-se, como sendo de Iluminação Global, os algoritmos *Ray-Tracing* e Radiosidade que se apresentam nas secções seguintes.

1.2.1 *Ray-Tracing*

O *Ray-Tracing* é um algoritmo de síntese de imagem extremamente versátil. Pode usar modelos de iluminação tão simples quanto o de Phong ou outros mais complexos, relacionados com os modelos físicos de reflexão e de transmissão de luz.

Appel desenvolveu um algoritmo designado por *Ray-Casting* [App68], destinado a resolver simultaneamente os problemas de cálculo de superfícies ocultas e de projecção de sombras. Um raio luminoso é emitido a partir do ponto de observação, através do centro de um *pixel*, para dentro da cena. Os pontos de intersecção entre o raio e os objectos em cena são calculados e o mais próximo do ponto de visão define a face visível nesse *pixel*. Para obter informação de sombra, o método emite um novo raio, designado por sensor de sombra (*shadow feeler*), para cada fonte de luz e com origem no ponto de intersecção. Se o sensor de sombra intersectar algum objecto, então o ponto está na sombra relativamente à fonte de luz respectiva.

Dado que a iluminação de um ponto de um objecto depende somente das fontes de luz, o algoritmo *ray-casting* é considerado ainda um algoritmo de iluminação local.

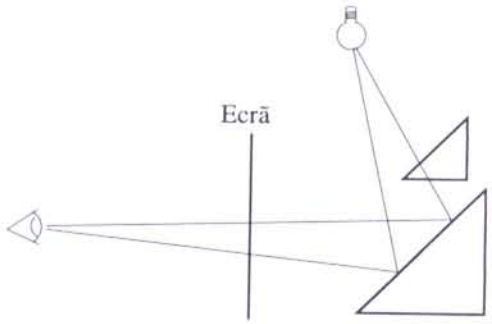


Fig. 1.6 - Algoritmo *ray-casting*

Whitted [Whi80], sugeriu um novo paradigma, mais geral do que o *ray-casting*, integrando no mesmo modelo a reflexão e a transmissão da luz, a remoção de superfícies ocultas e a projecção de sombras. Ao algoritmo resultante dá-se o nome *Ray-Tracing*.

O algoritmo *ray-tracing* segue recursivamente o percurso dos raios luminosos, por reflexão e por transmissão especulares. Um raio inicial é lançado a partir do ponto de observação, passando por um *pixel*, para dentro da cena. Uma vez determinado o objecto que o raio intersecta, lançam-se sensores de sombra em direcção às fontes de luz e geram-se dois novos raios, ambos com origem no ponto de intersecção: um raio segue na direcção de reflexão (definida por ângulos de incidência e de reflexão iguais) e outro na direcção de transmissão (segundo a Lei de Fresnel) se o objecto é transparente. O

algoritmo é recursivo visto que os dois novos raios são processados de uma forma idêntica ao raio inicial, dando origem a mais sensores de sombra e a mais raios reflectidos e transmitidos.

A figura 1.7 mostra o processo descrito de geração de raios. De forma a não sobrecarregar a figura, não se encontram representados os raios sensores de sombra.

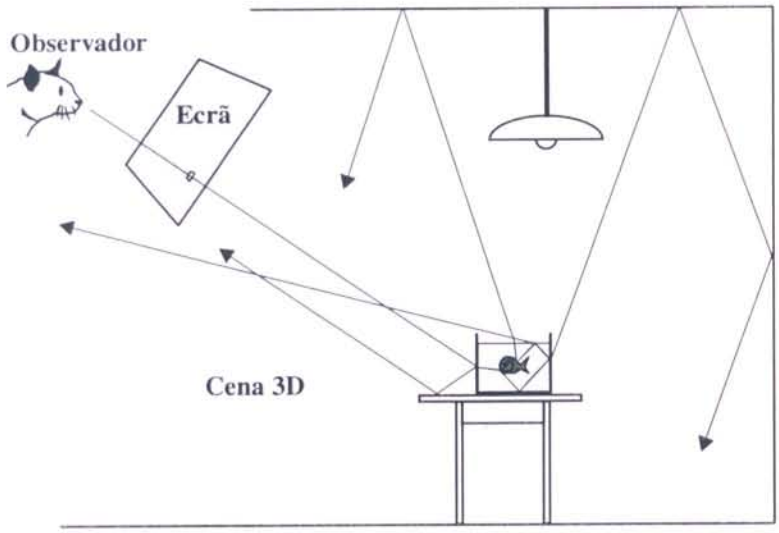


Fig. 1.7 - Exemplo de propagação de raios, em *ray-tracing*, a partir do mesmo raio inicial

Como resultado, o algoritmo constrói, para cada raio inicial, uma árvore de iluminação (*shading tree*) como a que se representa na figura 1.8. Os ramos da árvore representam raios reflectidos (R) ou transmitidos (T) e os nós representam os objectos intersectados. Os raios sensores de sombra (S) estão representados a partir dos nós da árvore, na horizontal.

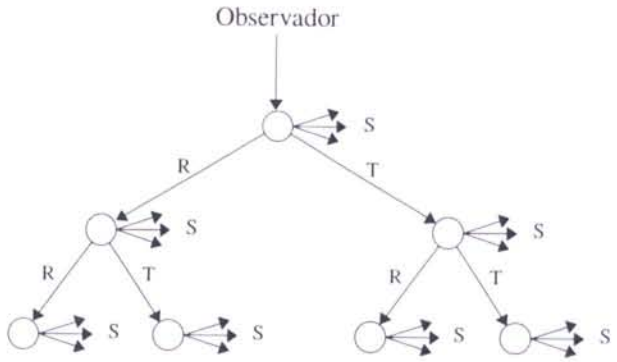


Fig. 1.8 - A árvore de iluminação em *ray-tracing*

O cálculo da cor a atribuir a um dado *pixel* efectua-se visitando a árvore de iluminação no sentido das folhas para a raiz, ao mesmo tempo que se avalia a intensidade em cada nó:

$$I(P) = I_{local}(P) + k_r \cdot I(P_r) + k_t \cdot I(P_t) \qquad \text{Eq. 1.3}$$

Onde:

P	intersecção no ponto em consideração;
P_r	primeira intersecção do raio reflectido a partir de P;
P_t	primeira intersecção do raio transmitido a partir de P;
$I_{local}(P)$	intensidade calculada por um modelo de iluminação local no ponto P;
$I(P)$	intensidade calculada recursivamente em cada ponto P;
k_r	coeficiente de reflexão;
k_t	coeficiente de transmissão.

Se o modelo de iluminação local utilizado é o de Phong, calcula-se a intensidade em cada ponto de intersecção, para cada comprimento de onda, adaptando a Eq. 1.3 de forma a acumular os efeitos das várias fontes de luz:

$$I_{local} = I_a k_a + \sum_i I_{fi} \cdot \left[k_d \cdot (N \cdot L_i) + k_r \cdot (N \cdot H_i)^n \right] \qquad \text{Eq. 1.4}$$

Com:

I_{fi}	intensidade da fonte de iluminação i;

L_i vector normalizado correspondente ao raio sensor de sombra lançado para a fonte i;

H_i vector normalizado que se situa a meio do ângulo formado pelos vectores L_i (para a fonte de luz) e V (para o observador)[1].

O algoritmo *ray-tracing* produz resultados razoavelmente realistas, principalmente no que diz respeito à simulação de efeitos especulares. A iluminação local é tratada de uma forma similar aos algoritmos anteriores [1.1], prevendo reflexão especular e difusa. A iluminação global é aproximada por somente dois raios, sendo um reflectido e o outro transmitido pela superfície de cada objecto intersectado. Sob o ponto de vista de iluminação global, este é o ponto fraco do *ray-tracing*. Fundamentalmente, a quantidade de pontos que influenciam a iluminação de um ponto particular reduz-se a dois, o ponto de intersecção do raio reflectido e o ponto de intersecção do raio transmitido. Efeitos directos desta aproximação são a ausência de cáusticas e a incapacidade de mistura de cores (*colour bleeding*).

A solução ideal passa pela criação de inúmeros raios transmitidos e reflectidos, em várias direcções, mas uma implementação da solução deve ser cuidadosamente pensada, pois pode tornar-se computacionalmente incomportável.

A generalização dos raios a cones (*Cone Tracing*, [Ama84]), pirâmides (*Beam Tracing*, [Hec84]), e famílias de raios vizinhos (*Pencil Tracing*, [Shi87]) procura ser uma solução para o problema.

J. Kajiya, em [Kaj86], apresenta um método que designa por *path tracing* que é uma boa aproximação à solução ideal. Em alternativa à geração de uma árvore completa de iluminação, com inúmeros descendentes por nó, cada ponto ou amostra no ecrã dá lugar a um percurso de raios. A direcção de cada raio reflectido (ou transmitido) é definida de acordo com a direcção do raio incidente, as propriedades da superfície reflectora (ou transmissora), e é desviada por um factor aleatório. Finalmente, a cor de um *pixel* é definida pela integração das cores das várias amostras que contém.

Ao integrar a informação de várias amostras aleatoriamente distribuídas dentro de um *pixel*, o *path tracing* resolve simultaneamente um outro problema normalmente associado com o *ray-tracing* que é o efeito de escada na imagem. Soluções alternativas para o problema, também elas baseadas numa amostragem estocástica são apresentadas em [Coo84], [Dip85], [Lee85], [Coo86], [Mit87]. Outras soluções subdividem

[1] O produto $N.H_i$ substitui o produto $R.V$ da Eq. 1.1, o que exige uma alteração do expoente n, mas tem a vantagem de exigir um processamento menos dispendioso.

recursivamente a área de um *pixel* e efectuam uma amostragem adaptativa [Whi80], [Cos89].

O algoritmo *ray-tracing*, tal como foi descrito, é conhecido por *eye ray-tracing*, devido à forma como os raios de luz são lançados a partir do observador, passando posteriormente pelos objectos, até chegarem às fontes de luz, ou seja, em sentido inverso ao real.

Pelo contrário, num algoritmo *light ray-tracing*[1], os raios são lançados a partir da fonte de luz, reflectem-se e/ou transmitem-se pelos objectos em cena e, alguns, atingem o ponto de observação. Como a quantidade de raios lançados é muito superior à quantidade de raios observados, efectuam-se muitos cálculos desnecessários o que tem dificultado a utilização de tais algoritmos. Refira-se no entanto que os resultados obtidos com algoritmos *light ray-tracing* tendem a ser melhores do que os obtidos com algoritmos *eye ray-tracing*, nomeadamente na simulação de cáusticas e de penumbras.

Uma implementação muito particular de *light ray-tracing* é apresentada em [Pat92]. Usa fotões em substituição de raios e recorre a técnicas de Monte Carlo. Os fotões são lançados estocasticamente a partir das fontes de luz e sempre que um fotão embate numa superfície decide-se, de acordo com as características de reflexão da mesma, se o fotão é absorvido ou reflectido, sendo, neste caso, processado recursivamente. O número de fotões que abandonam uma superfície, por emissão e/ou por reflexão, permitem determinar a intensidade de luz da superfície.

Um dos problemas mais graves que o algoritmo *ray-tracing* exibe é o tempo de processamento que consome na criação de uma imagem. Realmente, por cada amostra da imagem é gerada uma árvore de iluminação e, por cada raio que constitui esta última, é necessário visitar os objectos que compõem a cena 3D de forma a determinar o objecto cuja intersecção é mais próxima da origem do raio.

Vários trabalhos com o objectivo de acelerar o algoritmo *ray-tracing* têm sido publicados. Dado que a grande parte do tempo consumido pelo algoritmo corresponde a cálculos de intersecção de raios com objectos (95% para cenas complexas [Whi80]), a maioria desses trabalhos tenta reduzir a quantidade e a complexidade de testes de intersecção a efectuar.

---

[1] *Backward* e *forward ray-tracing* são designações alternativas para *eye ray-tracing* e *light ray-tracing* que têm sido preteridas nos últimos tempos por não existir consenso entre os vários autores. Para uns, *forward ray-tracing* corresponde a *light ray-tracing* por os raios lançados seguirem o sentido da luz; para outros corresponde a *eye ray-tracing* por os raios lançados seguirem o sentido especificado na versão original, apresentada por Turner Whited [Whi80]

Em [Kay86], propõem-se soluções baseadas na utilização de volumes envolventes dos objectos. Outros trabalhos baseiam-se na estruturação do espaço 3D como forma de acelerar as intersecções, seja por hierarquias de volumes envolventes [Gol87], seja por subdivisão do espaço [Gla84], [Fuj86].

Técnicas alternativas de aceleração de *ray-tracing* apoiam-se na direccionalidade dos raios para reduzirem o número de objectos candidatos à intersecção [Hai86], [Arv87] e na redução da quantidade de raios a processar [Hal83]. Também são conhecidas técnicas que aproveitam informação entre imagens consecutivas numa sequência [Seq89], [Mur90].

A paralelização do algoritmo é uma aceleração óbvia e será discutida no capítulo 2 deste documento.

A utilização das técnicas referidas não é no entanto suficiente para dar um carácter de interactividade ao algoritmo. Soluções que permitam uma antevisão da imagem final são necessárias, de forma que o utilizador seja capaz de avaliar, tão cedo quanto possível, as imagens que irá obter.

1.2.2 Radiosidade

O algoritmo Radiosidade tem origem nos métodos fundamentais do equilíbrio energético utilizados na engenharia térmica e foi introduzido na área da síntese de imagens, em Computação Gráfica, por Goral *et al.* [Gor84]. Calcula a radiosidade, ou seja, a energia por unidade de tempo e de área, em cada ponto dos objectos em cena.

Assume-se, no método, que os processos de emissão e de reflexão são difusos ideais (ou Lambertianos). Desta forma, a energia luminosa incidente numa superfície é reflectida com igual intensidade em todas as direcções.

Na base teórica do algoritmo está a noção dada anteriormente de iluminação global: a energia com origem numa uma área diferencial corresponde à soma da energia que emite com a energia que reflecte, oriunda de todas as outras áreas diferenciais que a rodeiam:

$$B_i dA_i = E_i dA_i + \rho_i \int_j B_j dA_j F_{dA_j\, dA_i} \qquad \text{Eq. 1.5}$$

Com:

dA_i área diferencial no ponto i considerado;

B_i radiosidade da área diferencial i;

E_i energia emitida pela área diferencial no ponto i por unidade de tempo e metro quadrado;

ρ_i reflectividade no ponto i;

dA_j área diferencial em cada um dos pontos j considerados;

B_j radiosidade de cada área diferencial j;

$F_{dAj\,dAi}$ factor de forma do ponto j para o ponto i, que representa a fracção de energia expelida por dA_j que atinge dA_i.

A Eq. 1.5 deve ser entendida como um método iterativo. A energia calculada $B_i dA_i$ é reenviada para a cena (ambiente fechado), influenciando o cálculo de outras $B_j dA_j$. Estas, por reflexão, influenciam novamente a energia $B_i dA_i$. Neste processo de reflexões consecutivas atinge-se um equilíbrio que corresponde à solução do problema, ou seja, a energia com origem em cada área elementar.

De forma a tornar tratável a Eq. 1.5, assume-se que a energia numa superfície é constante ao longo de toda a sua área. As superfícies são discretizadas em porções mais pequenas, designadas por *patches*. O integral da Eq. 1.5 é substituído por um somatório aplicado a todos os n *patches* em cena:

$$B_i A_i = E_i A_i + \rho_i \sum_{j=1}^{n} B_j A_j F_{ji} \qquad \text{Eq. 1.6}$$

A relação de reciprocidade entre os factores de forma de dois *patches* $F_{ij}A_i = F_{ji}A_j$ permite reescrever a Eq. 1.6:

$$B_i A_i = E_i A_i + \rho_i \sum_{j=1}^{n} B_j F_{ij} A_i \qquad \text{Eq. 1.7}$$

De onde se obtém a chamada equação de radiosidade:

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{ij} \qquad \text{Eq. 1.8}$$

Uma equação deste tipo existe para cada *patch* o que permite reescrevê-la sob a forma de um sistema de equações lineares:

$$B_i - \rho_i \sum_{j=1}^{n} B_j F_{ij} = E_i \qquad \text{Eq. 1.9}$$

Ou:

$$\begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \cdots & -\rho_2 F_{2n} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ -\rho_n F_{n1} & -\rho_n F_{n2} & \cdots & 1-\rho_n F_{nn} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \cdot \\ \cdot \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \cdot \\ \cdot \\ E_n \end{bmatrix} \qquad \text{Eq. 1.10}$$

Segundo esta equação, é possível calcular as radiosidades B_k dos *patches* da cena, uma vez conhecidos os valores das reflectividades ρ_k, dos factores de forma F_{ij} e das energias por unidade de tempo e de área E_k das fontes de luz.

A solução do sistema de equações anterior é independente do ponto de visão e a imagem é gerada por um algoritmo de visibilidade, vulgarmente um *z-buffer*. Nesta fase, a iluminação de cada *patch* é determinada, ponto a ponto, por meio de uma interpolação bilinear a partir da radiosidade nos seus vértices. Para algumas aplicações, o aspecto visual da imagem assim obtida é aceitável, dado que a interpolação bilinear garante a continuidade de grau C^0 entre *patches* vizinhos da mesma superfície. No entanto, a continuidade de grau superior não é garantido e, por esse motivo, a imagem é sujeita a efeito de *Mach band* [Fol90], [Wat92]. A discussão deste problema, assim como a descrição pormenorizada de uma solução, são efectuadas no capítulo 3 deste documento.

O sistema de equações anterior fornece os valores das radiosidades nos *patches*. As radiosidades nos vértices, necessárias à interpolação referida, são determinadas pela média das radiosidades dos *patches* que o partilham. Uma variante do método, apresentada por J. Wallace [Wal89], permite o cálculo de radiosidades directamente sobre os vértices.

A figura 1.9 resume os vários estágios de uma solução completa de um algoritmo de radiosidade, assim como os pontos onde o processo é retomado, caso haja lugar a alterações.

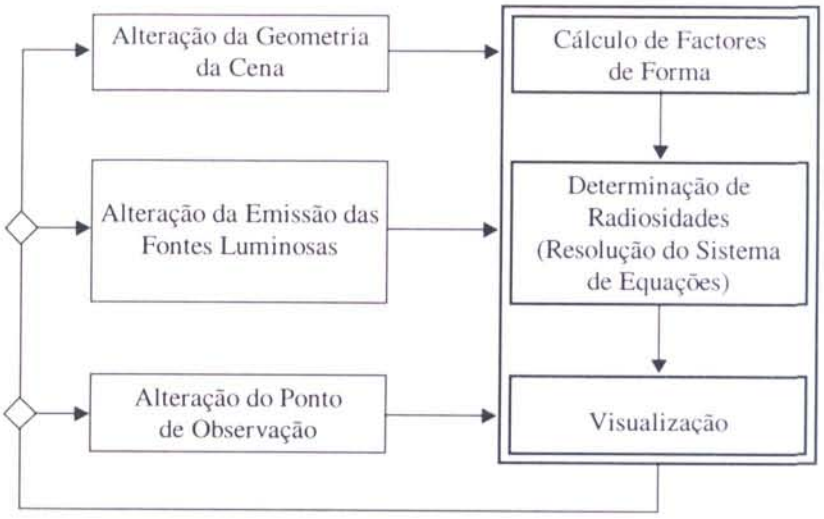


Fig. 1.9 - Os vários estágios na síntese de uma imagem segundo o algoritmo radiosidade

Dado que os cálculos efectuados são independentes do ponto de visão, uma alteração da posição do observador implica somente a repetição do bloco de visualização. Havendo alteração das características de reflexão das superfícies (ρ_k) ou das características das fontes de luz (E_k), é suficiente um nova resolução do sistema de equações lineares. Uma alteração da geometria da cena implica novos valores de factores de forma, pelo que todos os cálculos necessitam de ser repetidos. Trata-se da componente mais pesada do algoritmo.

1.2.2.1 Factores de Forma

Por definição, o factor de forma entre duas áreas elementares dA_i e dA_j corresponde ao quociente da energia que, partindo de dA_i, atinge dA_j, pela energia total que abandona dA_i em todas as direcções. Sendo ϕ_i e ϕ_j os ângulos formados respectivamente pelas normais de dA_i e de dA_j com a recta que une as duas superfícies elementares, o factor de forma entre duas superfícies elementares é calculável por:

$$F_{dA_i dA_j} = \frac{\cos\phi_i \cdot \cos\phi_j}{\pi \cdot r^2} \cdot dA_j \qquad \text{Eq. 1.11}$$

Da equação 1.11 podem ser deduzidas expressões para factores de forma de vértice (área elementar) para *patch* (F_{dAiAj}) e de *patch* para *patch* (F_{AiAj}):

$$F_{dA_i A_j} = \int_{A_j} \frac{\cos\phi_i \cdot \cos\phi_j}{\pi \cdot r^2} \cdot dA_j \qquad \text{Eq. 1.12}$$

$$F_{A_i A_j} = \frac{1}{A_i} \cdot \int_{Ai} \int_{Aj} \frac{\cos\phi_i \cdot \cos\phi_j}{\pi \cdot r^2} \cdot dA_j \cdot dA_i \qquad \text{Eq. 1.13}$$

A resolução dos integrais anteriores não é simples e daqui resulta que, a maior parte do (muito) tempo consumido pelo algoritmo radiosidade se deve ao cálculo dos factores de forma.

M. Cohen, em [Coh85], apresenta um método aproximado de cálculo para a solução do integral interno da equação 1.13 que designa por hemicubo. O hemicubo, colocado sobre a área diferencial, é dividido em *pixels* nos quais, por meio de um sistema *z-buffer*, projecta toda a cena. Uma vez fixados os factores de forma entre cada *pixel* do hemicubo e a área diferencial, o factor de forma de um qualquer *patch* e esta última calcula-se pelo somatório dos factores de forma dos *pixels* que contêm o *patch*. Se bem que, pela utilização de facilidades de *z-buffering* por *hardware*, o método seja relativamente eficiente, apresenta no entanto um problema grave, semelhante ao efeito de escada. Realmente, a precisão do método cresce com a resolução dada ao hemicubo e resoluções muito finas acabam por degradar a sua eficiência.

Outros métodos numéricos para o cálculo de factores de forma têm sido apresentados, parte dos quais são baseados na amostragem dos *patches*, por meio de lançamento de raios. Um desses métodos, apresentado por J. Wallace, em [Wal89], calcula factores de forma de *patch* para vértice e será apresentado no capítulo 3 deste documento.

1.2.2.2 Radiosidade com Refinamento Progressivo

A equação 1.10 corresponde a um sistema de equações, cuja resolução fornece a radiosidade de cada *patch* de uma cena 3D. Devido às dimensões que o sistema pode tomar, os métodos de resolução são normalmente do tipo iterativo.

O tipo de solução assim preconizado, baseado na equação 1.8, classifica-se vulgarmente como sendo de acumulação de energia (*gathering*) dado que, em cada iteração efectuada, um *patch* recolhe energia dos restantes.

Uma solução alternativa, apresentada em [Coh88], concentra-se no lançamento de energia (*shooting*) de um *patch* para os restantes, por cada iteração efectuada. O *patch*

emissor é o que, no momento, contém maior quantidade de energia para emitir. Os factores de forma não necessitam de ser conhecidos *à priori*, podendo ser calculados à medida que são necessários, o que permite eliminar a parcela de tempo inicial correspondente ao seu cálculo.

Esta solução é vulgarmente conhecida por Radiosidade com Refinamento Progressivo (*Progressive Refinement Radiosity*). A vantagem que exibe sobre a versão de acumulação é a possibilidade de se visualizar a imagem desde o início do cálculo, sendo os resultados refinados à medida que o processamento se efectua, o que corresponde, em termos da figura 1.9, a juntar os três estágios representados.

1.2.3 Combinação de *Ray-Tracing* e Radiosidade

Pelas descrições efectuadas nas secções anteriores acerca dos algoritmos *ray-tracing* e radiosidade, torna-se claro que uma combinação adequada dos dois algoritmos é uma via interessante para resolução do problema da iluminação global em toda a sua extensão, tal como é interpretada pela equação de *rendering* (Eq. 1.2).

Realmente, o algoritmo *ray-tracing* efectua uma boa aproximação das reflexões especulares, apresentando alguns problemas com as reflexões difusas, motivados pelo reduzido número de raios lançados de cada ponto de intersecção. Pelo contrário, o algoritmo radiosidade pressupõe que todas as superfícies são difusas ideais.

Em [Wal87], efectua-se um estudo das implicações da junção dos dois algoritmos, com base na formalização da transmissão de energia entre dois *patches*, segundo quatro vias: difuso para difuso, difuso para especular, especular para difuso e especular para especular.

Outros trabalhos, ditos de Inclusão de Especulares em Radiosidade, são conhecidos, vários dos quais se baseiam na junção dos dois algoritmos, *ray-tracing* e radiosidade. No capítulo 3, retomar-se-á este assunto.

1.3 Sumário

Neste capítulo apresentaram-se de forma sumária as várias vertentes da algoritmia relacionada com a síntese de imagem.

Na primeira parte, alguns algoritmos de visibilidade foram alvo de atenção. Dado que, nestes algoritmos, é difícil determinar a influência da iluminação de uns objectos sobre

os outros, classificaram-se como sendo algoritmos de Iluminação Local. Caracterizam-se por um nível baixo, tanto no custo computacional como no realismo das imagens geradas

Na segunda parte, definiu-se a noção de Iluminação Global, com base na interacção da iluminação dos vários objectos em cena. Neste contexto, foram apresentados dois algoritmos, *ray-tracing* e radiosidade, tendo-se acentuado as suas vantagens e desvantagens.

De facto, estes dois últimos algoritmos merecem uma atenção especial pois é sobre eles que recai grande parte dos trabalhos executados no âmbito deste documento.

O *ray-tracing* falha no processamento de superfícies difusas (problema potencialmente resolvido pela junção com radiosidade) e exibe um longo tempo de processamento. Nos capítulos 4 e 5 apresentar-se-ão soluções para a questão do tempo de processamento, baseadas na síntese de imagem com qualidade progressiva e na utilização de técnicas de paralelização.

Os problemas do algoritmo radiosidade são mais variados e bastante mais complexos. A reconstrução da função de iluminação numa superfície, a partir das radiosidades nos vértices e a junção de *ray-tracing* com radiosidade, aflorados neste capítulo, serão retomados com maior profundidade no capítulo 3.

Capítulo 2

Sistemas de Arquitectura Paralela em Síntese de Imagem

2. SISTEMAS DE ARQUITECTURA PARALELA EM SÍNTESE DE IMAGEM 2.1

2. Sistemas de Arquitectura Paralela em Síntese de Imagem

Nos últimos tempos, tem-se assistido a uma evolução sensível no desempenho dos computadores digitais com arquitecturas tradicionais. No entanto, a necessidade de máquinas cada vez mais rápidas tem favorecido a investigação e o desenvolvimento de arquitecturas especiais, capazes de responder às necessidades mais exigentes dos utilizadores.

Neste contexto, as arquitecturas paralelas de computadores são uma via importante a considerar, tanto em sistemas de utilização genérica, como em sistemas específicos.

Reservadas inicialmente a máquinas de médio e de grande porte, as arquitecturas paralelas sofreram um impulso significativo durante a década de 80, motivado pela comercialização de processadores especialmente desenvolvidos ou adaptados para o efeito. O termo Processamento Massivamente Paralelo (*massively parallel processing*), passa então a ser mais vulgar e hoje em dia descrevem-se máquinas com centenas e mesmo milhares de processadores.

Um desses processadores é o *Transputer* [Inm92]. À data do seu aparecimento, o desempenho individual e as facilidades de comunicação que apresenta, levam a que seja amplamente utilizado, mesmo nos meios mais vocacionados para a investigação. Alguns sistemas dedicados, nomeadamente para a síntese de imagens, têm vindo a ser apresentados desde então. Alguns trabalhos referidos nos capítulos seguintes são

desenvolvidos com base em *Transputers* e em linguagem de programação OCCAM, considerada importante no contexto geral destes processadores.

Neste capítulo efectua-se uma visita a alguns conceitos da Computação Paralela, com incidência nas Arquitecturas e nos Modelos de Programação e descreve-se o conjunto de parâmetros vulgarmente utilizados para a Avaliação do Desempenho dos Sistemas Paralelos. Passa-se à descrição do *Transputer*, sob o ponto de vista da arquitectura e do modelo de programação e apresentam-se alguns rudimentos da linguagem OCCAM, essenciais para a boa compreensão de alguns alguns algoritmos apresentados nos capítulos 4 e 5.

O capítulo termina com a apresentação de algumas implementações do algoritmo *ray-tracing* em sistemas paralelos. Pretende-se aqui identificar os principais problemas e vantagens das várias soluções que preconizam, com vista a obter-se uma perspectiva do estado da arte, fundamental para a boa compreensão dos capítulos 4 e 5.

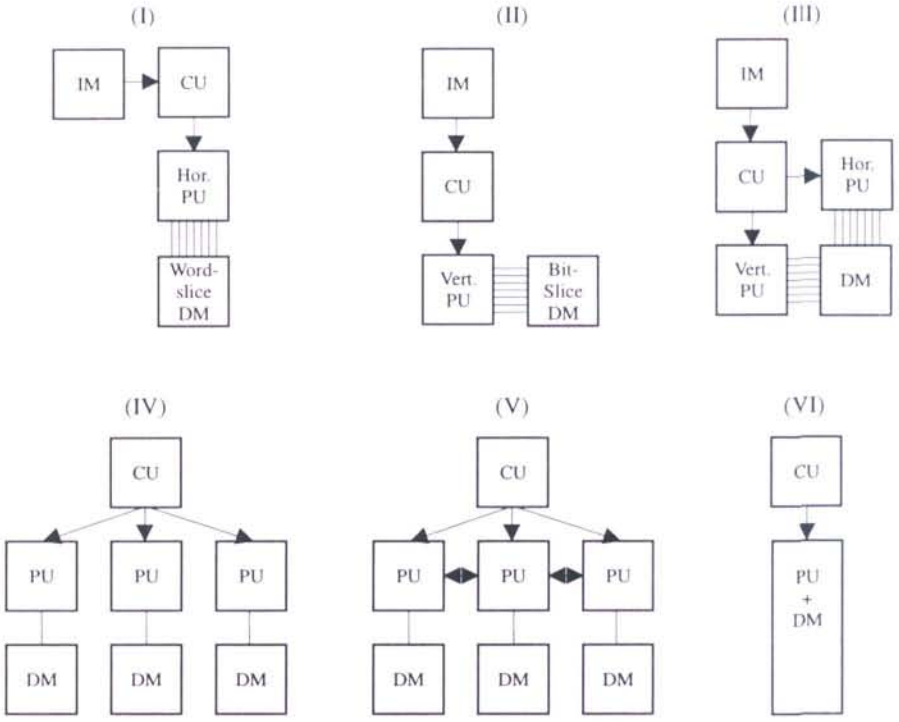
2.1 Arquitecturas Paralelas de Computadores

R. Hockney e C. Jesshope, em [Hoc81], referem duas taxonomias diferentes para as arquitecturas paralelas de computadores, a de Flynn e a de Shore.

Na taxonomia de Shore reconhecem-se seis classes de máquinas paralelas de acordo com a organização das suas partes constituintes: Máquina I a Máquina VI (figura 2.1).

A Máquina I corresponde a uma máquina sequencial, na qual uma operação de leitura na memória de dados (DM) fornece todos os *bits* de uma palavra para processamento em paralelo pela unidade de processamento (PU), sob a orientação de uma unidade de controlo (CU). Na Máquina II, uma operação de leitura extrai uma coluna de um *bit* de várias palavras consecutivas e a unidade de processamento é organizada de modo a operar num formato *bits* em série. A Máquina III corresponde a uma combinação das duas anteriores. As Máquinas IV e V usam uma única unidade de controlo para várias unidades de processamento com memórias de dados próprias, diferindo entre si pela capacidade de comunicação entre as unidades de processamento. A Máquina VI pretende distribuir a lógica da unidade de processamento pela própria memória.

Algumas dificuldades na aplicação da taxonomia para classificação de máquinas reais têm-na preterido a favor da taxonomia de Flynn que continua a ser referida por todos os autores de especialidade.



IM	Memória de Instruções
DM	Memória de Dados
CU	Unidade de Controlo
PU	Unidade de Processamento

Fig. 2.1 - As seis categorias da taxonomia de Shore

A taxonomia de Flynn dá menos importância à arquitectura da máquina para se concentrar nos tipos de cadeias (*streams*) de instruções e de dados. Uma cadeia pode ser simples (*single*) ou múltipla, do que resulta uma classificação em quatro classes de máquinas.

SISD - *Single Instruction stream/Single Data stream*: corresponde à máquina sequencial de von Neumann; uma instrução é descodificada numa unidade de tempo e a cadeia de instruções manipula uma cadeia de dados.

MISD - *Multiple Instruction stream/Single Data stream*: das quatro classes é a menos intuitiva. Configurações possíveis baseiam-se numa cadeia de processadores, segundo a qual os dados flúem de processador em processador, sendo modificados em cada um deles. Em [Qui94], enquadram-se nesta interpretação os Vectores Sistólicos de processadores, enquanto que em [Sch88] tais configurações tomam o nome de *Macropipeline*.

SIMD - ***Single Instruction stream/Multiple Data stream***: uma operação é desencadeada simultaneamente em várias unidades de processamento, cada uma das quais acede a diferentes cadeias de dados.

MIMD - ***Multiple Instruction stream/Multiple Data stream***: cada elemento de processamento executa uma cadeia de instruções diferente, sobre cadeias de dados também diferentes. Nos capítulos 4 e 5 serão aprsentados trabalhos desenvolvidos sobre uma arquitectura deste tipo.

O termo Vector de Processadores significa um computador SIMD implementado como uma máquina sequencial (*front end*) ligada a um conjunto de elementos de processamento idênticos e sincronizados, capazes de executar simultaneamente a mesma operação em diferentes dados (figura 2.2). Difere de um Processador Vectorizado em *Pipeline* que fornece vectores de dados da memória para o CPU, onde um *pipeline* de unidades aritméticas os manipula. Ambos os termos se incluem num outro mais abrangente, o Computador Vectorizado (*Vector Computer*).

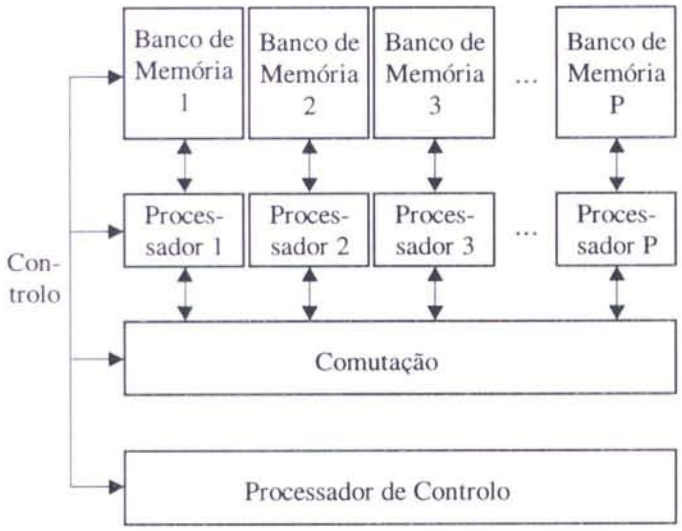


Fig. 2.2 - Arquitectura de um vector de processadores

O termo Multiprocessador significa um computador MIMD constituído de vários processadores, com memória partilhada, cada um dos quais capaz de executar o seu próprio programa. Pelo facto de a memória ser partilhada, a informação num processador é facilmente acedida por outro processador e, por este motivo, os multiprocessadores são também conhecidos por sistemas MIMD Fortemente Ligados.

Dependendo da memória partilhada ser centralizada ou distribuída, o multiprocessador diz-se do tipo UMA - *Uniform Memory Access* ou NUMA - *Non Uniform Memory Access* respectivamente.

Numa configuração UMA, a memória é acedida por todos os processadores de igual forma, através de um mecanismo de comutação adequado que pode ser um barramento comum, um *crossbar switch* ou uma rede *packed-switch* (figura 2.3).

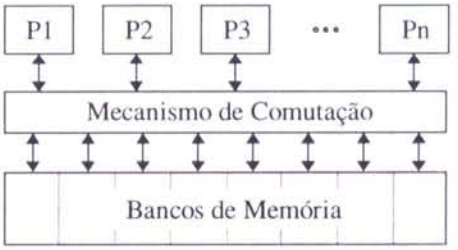


Fig. 2.3 - Multiprocessador do tipo UMA

Numa configuração NUMA, cada processador tem a sua própria memória local e o espaço partilhado de memória é composto pela combinação dos vários espaços locais (figura 2.4). O tempo de acesso de um processador a um endereço local é menor do que o tempo de acesso a um endereço remoto.

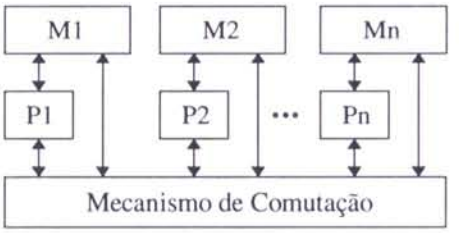


Fig. 2.4 - Multiprocessador do tipo NUMA

O termo Multicomputador é usado para designar um computador MIMD constituído de vários processadores, sem memória partilhada, que interagem através da comunicação de mensagens. Os multicomputadores são também conhecidos por sistemas MIMD Fracamente Ligados.

Dada a dificuldade de efectuar todas as ligações necessárias à comunicação entre qualquer par de processadores, uma mensagem é obrigada a passar por vários processadores intermédios antes de atingir o seu destino.

As arquitecturas mais avançadas de multicomputadores associam, a cada processador, *hardware* especializado que permite a passagem de mensagens através do nó, sem necessidade de memorização das mensagens e sem interrupção da unidade de processamento respectiva.

Em implementações mais simples, a passagem de mensagens é efectuada com base na estratégia recebe, memoriza e envia (*store and forward message passing*): à medida que o nó recebe a mensagem, coloca-a em memória; se a mensagem se destina a outro nó, identifica o encaminhamento a dar-lhe e lê-a da memória à medida que a envia pela via de saída correspondente.

São importantes, neste caso, os mecanismos de acesso directo a memória (DMA) por libertarem as unidades de processamento dos nós intermédios das tarefas de comunicação. A sua interrupção continua no entanto a ser necessária para iniciar a recepção ou a emissão de mensagens.

2.2 Mecanismos de Decomposição de Problemas

Face à implementação de um problema numa arquitectura paralela, é necessário encontrar as diferentes formas de paralelismo que o problema eventualmente possa exibir. Interessa, fundamentalmente, distribuir equitativamente a carga computacional do problema pelos vários processadores (*Load Balancing*).

Os mecanismos de fluência de programas (*program flow mechanisms*) genericamente aceites são mencionados por K. Hwang em [Hwa93]:

- Mecanismos de Fluência
  - *Control Flow*
  - *Data Flow*
  - *Reduction*
    - *String Reduction*
    - *Graph Reduction*

Por *control flow*, entende-se um mecanismo do tipo do que se encontra num computador puramente sequencial, ou seja, a ordem pela qual as operações são executadas é explicitamente expressa pelo programador. A diferença numa arquitectura paralela é que devem ser encontrados, à priori, os segmentos de programa cujas dependências permitam a sua execução em paralelo.

Num mecanismo *data flow* não existe, teoricamente, uma ordem de execução das instruções que compõem um programa, isto é, todas as instruções estão prontas a ser executadas assim que os dados necessários à sua execução estejam disponíveis.

Num mecanismo do tipo *reduction*, as várias operações vão sendo decompostas em outras menores. Tal decomposição pode ser do tipo *string reduction*, se efectuada por recursão (uma operação fica suspensa enquanto os seus argumentos de entrada vão sendo calculados e diz-se completamente reduzida quando todos os argumentos forem substituídos pelos respectivos valores) ou do tipo *graph reduction* se efectuada por decomposição de um grafo dirigido que representa o conjunto de operações (os subgrafos são processados tanto quanto possível em paralelo).

Alan Chalmers, em [Cha91], apresenta uma taxonomia da decomposição de problemas mais adaptada a máquinas de arquitectura MIMD:

- Decomposição
  - Algorítmica
    - *Fork & Join*
    - *Data Flow*
  - No Domínio
    - Orientada aos Dados
    - Orientada às Tarefas

As decomposições algorítmicas *Fork & Join* e *Data Flow* correspondem respectivamente às decomposições *Control Flow* e *Data Flow* mencionadas por Hwang.

Por Decomposição no Domínio, entende Chalmers uma decomposição que explora o potencial paralelismo que existe na aplicação simultânea do mesmo algoritmo a diferentes itens de dados (cada processador contém uma réplica do algoritmo). Define tarefa como sendo a aplicação do algoritmo a um conjunto restrito de dados e distingue dados específicos da tarefa e outros dados.

Se cada processador tem acesso à globalidade de outros dados, então o trabalho distribui-se pelos vários processadores atribuindo-lhes tarefas à medida que ficam disponíveis e a decomposição diz-se Orientada às Tarefas. Se pelo contrário, cada processador contém um conjunto restrito de outros dados aos quais aplica o algoritmo em questão, a decomposição é Orientada aos Dados.

Algumas considerações devem ser feitas quanto à eficiência e usabilidade de cada um dos modelos de decomposição enunciados.

O modelo *reduction* apresenta como vantagens um elevado potencial de paralelismo e uma manipulação fácil das estruturas de dados. Como principal desvantagem, o tempo de propagação de pedidos de cálculo.

O modelo *data flow* apresenta um elevado potencial de paralelismo, permitindo mesmo uma granularidade muito fina, ao nível da instrução. No entanto, requer mecanismos especiais, complexos, para detectar a disponibilidade de dados necessários ao início de um processo, o que acarreta *overheads* significativos.

O modelo *fork & join* (*control flow* segundo Hwang) usa memória partilhada para suportar instruções de programa e itens de dados, o que tem a vantagem de permitir um controlo absoluto sobre a evolução de um programa e sobre estruturas de dados complexas. No entanto, a obrigatoriedade da existência de memória partilhada torna a sua aplicação limitada a certas arquitecturas.

Por exclusão de partes, os modelos de decomposição no domínio apresentam-se assim como os mais indicados para a programação de sistemas MIMD de memória distribuída.

O modelo de decomposição no domínio e orientado aos dados apresenta algumas dificuldades em distribuir equitativamente a carga computacional pelos vários processadores. Realmente, a complexidade associada aos conjuntos de dados distribuídos pelos vários processadores pode ser extremamente variável. Se os dados são distribuídos estaticamente, grandes diferenças resultam nos tempos de cálculo correspondentes à aplicação do mesmo algoritmo sobre conjuntos diferentes de dados, deixando alguns processadores sem trabalho enquanto outros ficam com trabalho excedente. Se os dados são redistribuídos dinamicamente, com base nas cargas computacionais de cada processador, então a comunicação entre processadores tende a aumentar fortemente e a eficiência diminui por esse motivo.

O modelo de decomposição no domínio e orientado às tarefas mostra-se mais eficiente a equilibrar as cargas computacionais dos vários processadores. Sempre que um processador termina uma tarefa, fica em estado inactivo e pede uma nova tarefa ao sistema. Os problemas principais do modelo relacionam-se com a complexidade das tarefas versus comunicação entre processadores e com a disponibilidade dos dados necessários à execução de cada tarefa.

Para se obter um bom equilíbrio de cargas computacionais, dever-se-ão definir tarefas de tamanho reduzido (granularidade fina). Este procedimento faz aumentar as comunicações entre processadores e, consequentemente, induz o aparecimento de *overheads* que diminuem a capacidade de processamento da arquitectura. O dimensionamento das tarefas, ou da sua granularidade deve ser portanto balanceado com as comunicações de forma a obter um bom compromisso.

Em problemas de grande complexidade, o conjunto de dados necessários à resolução de um determinado problema ocupa uma área de memória excessiva. Numa máquina MIMD

de memória distribuída, é importante criar os meios necessários para disponibilizar em cada processador, a baixo custo, os dados necessários à execução de cada tarefa.

2.3 Medida do Desempenho dos Sistemas Paralelos

Para uma dada implementação numa arquitectura paralela, interessa obter informação acerca da qualidade da paralelização efectuada. As duas principais grandezas que caracterizam uma implementação paralela são o *Speedup* e a Eficiência.

2.3.1 *Speedup*

Entende-se por *speedup* de uma paralelização, o quociente do tempo necessário para resolver um dado problema com um só processador pelo tempo necessário para resolver o mesmo problema com *Np* processadores:

$$Sup = \frac{T_1}{T_{Np}} \qquad \text{Eq. 2.1}$$

O tempo T_1 pode ser medido de duas formas: fazendo correr num processador da mesma máquina o algoritmo sequencial tido como o mais eficiente ou uma versão do algoritmo paralelo em estudo configurada para um processador.

Não parece justo comparar tempos de versões completamente sequenciais com tempos de versões paralelas, principalmente quando se trata de sistemas massivamente paralelos. Além de questões meramente algorítmicas, existem outras, de carácter marcadamente arquitectural que, introduzem erros na avaliação de T_1, umas por excesso[1] e outras por defeito[2]. Uma discussão deste assunto pode ser encontrada em [Qui94] que substitui o termo *speedup* por *parallelizability* para a situação de T_1 ser medido com uma versão paralela do algoritmo num processador. O termo *speedup* usado neste trabalho corresponde ao termo *parallelizability*.

Na situação ideal, é suposto que um incremento do número de processadores produz um incremento, na mesma relação, na velocidade de processamento de um problema. Esta

[1] Por exemplo devido à diminuição de memória endereçável, obrigando a mais frequentes chamadas ao sistema operativo.
[2] Por exemplo devido aos *overheads* de comunicações, inexistentes numa versão sequencial.

situação é designada por *speedup* linear e corresponde, graficamente, a uma linha recta de inclinação 1 conforme se mostra na figura 2.5.

Em condições normais, a situação ideal não é atingível dado que um algoritmo não é completamente paralelizável. Todo o programa paralelo tem certos custos de *overhead*, tais como criação de processos, comunicações, sincronização, etc. Estes custos podem ser vistos como componentes do algoritmo que se executam de uma forma sequencial e influenciam a qualidade da paralelização.

A Lei de Amdahl [Amd67] explica a forma como as componentes sequenciais de um algoritmo paralelo limitam o *speedup* máximo do sistema: seja uma arquitectura paralela, constituída por N_p processadores e executando um programa cujas componentes puramente sequencial e paralelizável consomem respectivamente s e p fracções do tempo total de processamento num processador ($s+p=1$); substituindo na Eq. 2.1, obtém-se a expressão do valor máximo de *speedup*:

$$MaxSup = \frac{s+p}{s+\frac{p}{N_p}} = \frac{1}{s+\frac{p}{N_p}} \qquad \text{Eq. 2.2}$$

Segundo a Lei de Amdahl, o *speedup* máximo que se pode obter de um sistema paralelo quando o número de processadores tende para infinito é $1/s$. Se, por exemplo, as componentes sequenciais consomem 1% do tempo de processamento total, o *speedup* máximo é 100, ainda que sejam utilizados muitos mais processadores. Esta situação encontra-se representada na figura 2.5, na qual algumas curvas de *speedup*, para várias dimensões m do problema a resolver, são comparadas com a recta de *speedup* ideal.

Pode observar-se, na figura, que um valor máximo das curvas de *speedup* se obtém para um determinado número óptimo de processadores. Esta observação contraria, de certa forma, os resultados da equação 2.2. Realmente, a componente puramente sequencial s não é constante e tende a aumentar à medida que o número de processadores aumenta[1], o que justifica a observação anterior.

A Lei de Amdahl analisa o comportamento dos sistemas paralelos, com um número crescente de processadores, mas mantendo fixa a dimensão do problema. Fazendo variar a dimensão m de um dado problema, pode verificar-se o chamado Efeito de Amdahl,

[1] Por exemplo, num sistema multicomputador, o aumento de processadores implica, normalmente, um maior fluxo de mensagens entre processadores. O fluxo de mensagens afecta a componente sequencial, dado que, através de um mesmo canal, a transmissão de mensagens não é paralelizável.

segundo o qual, à medida que a dimensão m de um problema cresce, a componente sequencial s decresce percentualmente e, em consequência, as curvas de *speedup* tendem a aproximar-se da recta ideal (figura 2.5).

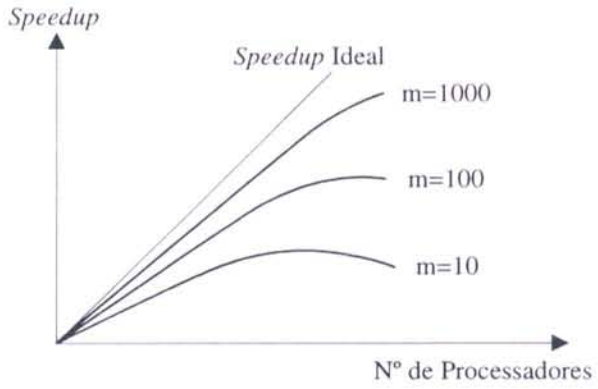


Fig. 2.5 - Curvas de *speedup* segundo a Lei de Amdahl

J. Gustafson, em [Gus88], contesta o pessimismo traduzido pela Lei de Amdahl no que respeita à fixação da dimensão do problema. Segundo o autor, "a dimensão de um problema cresce proporcionalmente com o número de processadores".

Com este pressuposto e, mais uma vez, com a suposição de invariabilidade da componente sequencial s, Gustafson conclui que o que passa a designar por *Scaled Speedup* tem uma evolução com o número de processadores muito mais próxima do *speedup* ideal:

$$ScaledSup = N_P + (1 - N_P) \cdot s \qquad \text{Eq. 2.3}$$

A evolução do *scaled speedup* com o número de processadores é apresentada na figura 2.6.

X. Sun e L. Ni, em [Sun93], efectuam uma análise mais profunda do comportamento de sistemas paralelos, incluindo o factor quantidade de memória. Os contributos de Amdahl e Gustafson podem ser vistos, à luz do modelo concebido por aqueles autores, como casos particulares. Segundo o modelo de memória constante de Sun e Li, quando a necessidade de cálculo aumenta mais rapidamente do que a necessidade de memória, pode obter-se um *speedup* ainda superior ao de Gustafson.

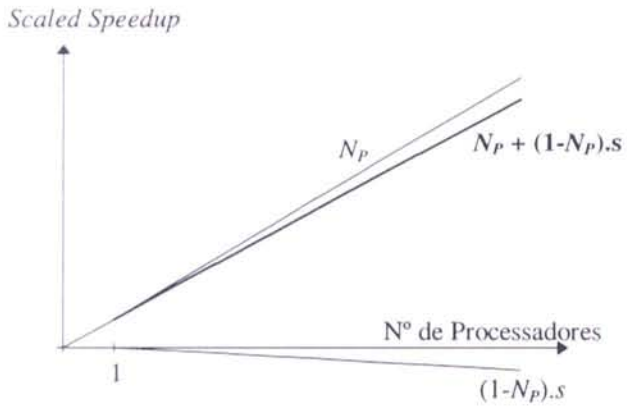


Fig. 2.6 - Evolução do *scaled speedup* segundo Gustafson

2.3.2 Eficiência

O parâmetro eficiência dá uma ideia clara, em termos percentuais, da utilização dos N_p processadores. Uma eficiência de 80% significa que, em termos médios, cada processador está inactivo ou ocupado com *overheads*, durante 20% do tempo total de processamento.

$$Eficiencia = \frac{Sup}{N_p} \cdot 100 \qquad \text{Eq. 2.4}$$

Se o *speedup* do sistema é o ideal (recta de inclinação 1), então a eficiência tem um valor independente do número de processadores, 100%. Se o *speedup* tem um evolução linear abaixo do ideal (inclinação inferior a 1), a eficiência mantém um valor constante mas inferior a 100%. Se o *speedup* tem uma evolução sublinear, então a eficiência baixa à medida que o número de processadores aumenta.

2.4 O *Transputer*

O *Transputer* é um processador especialmente desenvolvido para a implementação de arquitecturas paralelas do tipo MIMD, com memória distribuída (multicomputadores).

A sua arquitectura interna define uma família de componentes VLSI programáveis. Tipicamente, cada membro da família compõe-se de uma unidade central de

processamento, memória e meios físicos de comunicação com o exterior [Inm92]. É portanto lícito designar um *transputer* por Microcomputador.

Dos vários componentes da família, que foram sendo comercializados desde o seu aparecimento, no início da década de 80, o T800 (e seus descendentes directos, T801 e T805) é o mais utilizado. Salvo afirmação em contrário, o presente texto refere-se a este componente. Mais recentemente e seguindo uma filosofia semelhante, foi comercializado o último componente da família, com a referência T9000. Este apresenta, além de um melhor desempenho, mais facilidades ao nível das comunicações.

O *transputer* T800 inclui, no mesmo circuito VLSI, uma unidade central de processamento de 32 *bits*, uma unidade de vírgula flutuante por *hardware* capaz de efectuar operações aritméticas a 32 e a 64 *bits* (segundo a norma IEEE 754), uma memória estática rápida com uma capacidade de 4 K*Bytes*, uma interface para memória exterior com uma capacidade de endereçamento de 4 G*Bytes*, uma interface de comunicações com os quatro *Links*, dois relógios (*timers*) com resoluções de 1 e 64 μseg e uma unidade de serviços para sistema (figura 2.7).

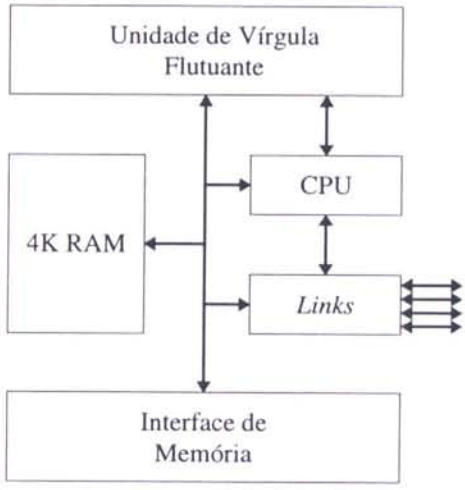


Fig. 2.7 - Arquitectura interna de um *transputer*

Os *Links* constituem a chave da construção de sistemas paralelos baseados em *transputers*. Cada *link* é fisicamente constituído por duas linhas, de forma a possibilitar a comunicação série nos dois sentidos, podendo funcionar a uma taxa de 20 M*Bits* por segundo. São dotados de capacidade de acesso directo a memória (DMA) o que lhes permite a realização de comunicações em paralelo com a unidade central de processamento e com as outras unidades da arquitectura do processador.

Cada *link* efectua a ligação ponto a ponto entre dois quaisquer componentes da família, independentemente da frequência e da fase do ciclo de relógio de cada um. Desta forma é possível concretizar multicomputadores com as mais variadas topologias de rede (figura 2.8).

A vocação do *transputer* para o processamento paralelo começa nas facilidades que oferece para o processamento em concorrência. Para o efeito, um *scheduler* integrado no *hardware* mantém duas listas de processos activos[1], de acordo com duas classes de prioridade: os processos de alta prioridade correm até terminarem ou até ficarem suspensos numa operação de entrada/saída; aos processos de baixa prioridade são atribuídas fatias de tempo, sempre que estejam prontos a correr e não existam processos de alta prioridade nessa situação. Os vários processos comunicam entre si através de canais implementados em *software*.

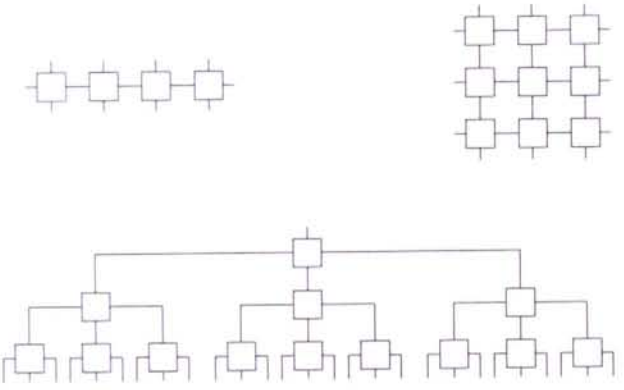


Fig. 2.8 - Topologias de redes de *transputers* em linha, em matriz 2D e em árvore ternária

A unidade central de processamento constitui-se de seis registos de 32 *bits* (figura 2.9). Os registos A, B e C, são de uso genérico e organizam-se em pilha. Os registos *Workspace* e *Next Instruction* contêm apontadores para áreas de memória relacionadas com o processo em curso, respectivamente, área de memória que contém variáveis locais e próxima instrução a ser executada. No *Operand Register* reconstruem-se os operandos das instruções.

[1] Um processo activo está em execução ou em lista de espera. Um processo inactivo aguarda o aparecimento de um evento (entrada/saída, valor de relógio).

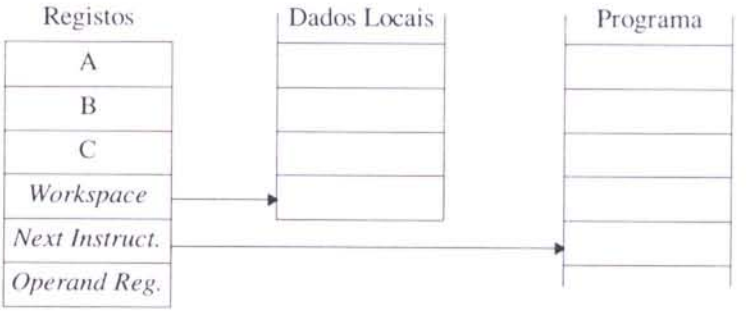


Fig. 2.9 - Os registos do *transputer*

A codificação de instruções é feita preferencialmente num único *byte*, dividido em dois grupos de quatro *bits*: o mais significativo para o código da instrução e o outro para o respectivo operando [Inm88a].

Quando uma instrução requer um operando de comprimento superior a quatro *bits*, uma instrução prefixo garante a leitura das várias fatias que o compõem e que, no *operand register*, vão sendo deslocadas à esquerda e concatenadas até à total composição do operando.

Nos dezasseis valores que correspondem aos quatro *bits* mais significativos de um código de instrução, codificam-se as treze instruções mais utilizadas (seguindo uma filosofia próxima da utilizada nos processadores RISC - *Reduced Instruction Set Computer* [Gim89]), dois prefixos para composição de operandos e um prefixo para funções indirectas que permite a codificação de um grande conjunto de instruções, com flexibilidade suficiente para permitir a sua expansão.

2.5 Uma Linguagem de Programação Paralela: OCCAM

Nesta secção pretende-se dar uma visão geral sobre a linguagem OCCAM, visitando as suas definições e construções principais, mesmo que sem grande rigor sintáctico e semântico. O objectivo é fornecer os fundamentos básicos necessários à compreensão dos algoritmos apresentados nos capítulos posteriores.

A linguagem OCCAM1 foi a primeira linguagem de programação utilizada no desenvolvimento de sistemas baseados em *transputers* [Jon87]. Bastante rudimentar, rapidamente deu lugar ao aparecimento da linguagem OCCAM2, mais poderosa, nomeadamente na incorporação de aritmética de vírgula flutuante. No decorrer deste

trabalho, a referência a OCCAM será, na maioria das situações, relativa à segunda versão da linguagem.

Transputer e OCCAM foram desenvolvidos em simultâneo, existindo uma influência directa da linguagem sobre o modelo de programação do processador: linguagem e modelo de programação são semelhantes [May89b], [May89c]. Por isso, a linguagem OCCAM é muitas vezes vista como a linguagem intrínseca à programação do *transputer*, havendo mesmo quem a defina como sendo o seu *Assembly*.

Em OCCAM, os vários blocos que compõem um programa são designados por processos. Cada processo inicia-se, executa uma série de acções e termina. Os processos podem ser executados em sequência ou em paralelo.

Dada a similaridade do modelo de programação do *transputer* com a linguagem OCCAM, os processos em paralelo podem ser executados em simultaneidade absoluta, se colocados em processadores diferentes, ou em concorrência, se colocados no mesmo processador. Salvaguardadas as condicionantes de quantidade de memória, o mesmo conjunto de processos pode ser igualmente mapeado em processadores diferentes ou no mesmo processador (figura 2.10).

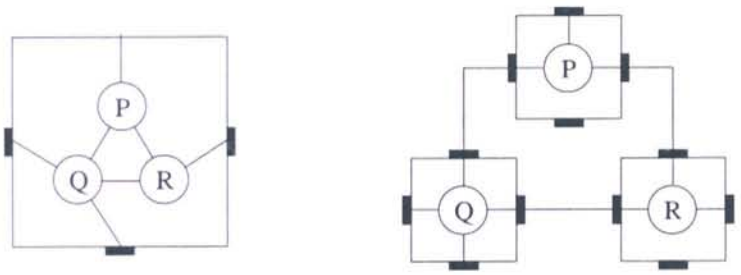


Fig. 2.10 - Um conjunto de processos pode ser mapeado no mesmo processador ou em processadores diferentes

A comunicação entre processos é efectuada através de mensagens passando através de canais. Cada canal é uma ligação ponto a ponto, não memorizada (*unbuffered*), o que significa que a comunicação só se realiza quando ambos os processos se encontram prontos a comunicar. Se os processos em comunicação se encontram em processadores diferentes, os canais são implementados fisicamente sobre os *links* do *transputer* (canais *hardware*). Se os processos se encontram no mesmo processador, o canal corresponde a uma cópia de memória para memória (canal *software*).

Apresenta-se de seguida um resumo das facilidades disponibilizadas pelas linguagem de programação OCCAM. Uma descrição mais completa pode ser encontrada em [Pou87] e [Inm88b].

Os processos elementares em OCCAM são a atribuição (p.e. x := y), o *input* de um canal ou de um relógio (p.e. ch ? x) e o *output* para um canal (p.e. ch ! x). Os processos podem ser agrupados em construções. As construções são a Sequência (SEQ), o Paralelo (PAR), o *Input* Alternativo (ALT), o Ciclo (WHILE), a Selecção (CASE) e o Condicional (IF).

A construção SEQ define um conjunto de processos a executar sequencialmente, pela ordem de escrita, enquanto que a construção PAR define um conjunto de processos a executar em paralelo. Na figura 2.11, os processos P2 e P3 são executados em sequência mas em paralelo com o processo P4. Globalmente, executam-se em sequência o processo P1, os processos englobados em PAR e o processo P5. Note-se que o processo P5 só será iniciado quando todos os processos da construção PAR tiverem terminado.

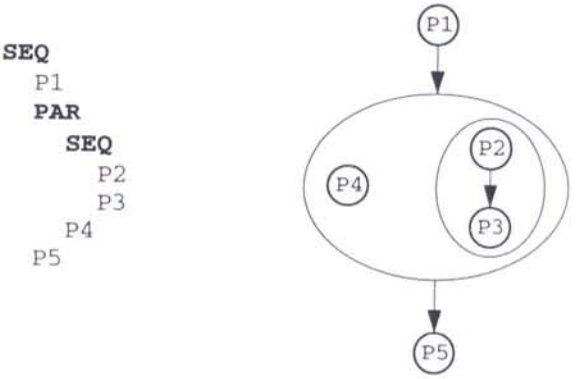


Fig. 2.11 - Exemplo de construções SEQ e PAR em OCCAM

As construções CASE e WHILE não diferem muito das equivalentes encontradas noutras linguagens. A sintaxe de ambas apresenta-se na figura 2.12.

```
WHILE <condição>
  <processo>
```

```
CASE <expressão>
  <lista de constantes>
    <processo_0>
  <lista de constantes>
    <processo_1>
  ...
  ELSE
    <processo_n>
```

Fig. 2.12 - Construções CASE e WHILE em OCCAM

A construção IF permite incluir, na mesma instrução, a avaliação de várias condições sob a forma de expressão Booleana, pela ordem de escrita. A primeira condição encontrada pela ordem de escrita, cujo resultado é verdadeiro, inicia um processo e a construção IF termina. Se não se verificar nenhuma das condições presentes, a construção entra em estado de STOP e, por esta razão, é vulgar que a última condição seja uma expressão com resultado garantidamente verdadeiro. Na figura 2.13 apresenta-se a sintaxe da construção IF e um exemplo. A última condição do exemplo, TRUE, garante que a construção não entra em situação de STOP.

```
IF
  <condição_0>
    <processo_0>
  <condição_1>
    <processo_1>
  <condição_2>
    <processo_2>
  ...
```

```
IF
  x > 0
    y := y + x
  x < 0
    y := y - x
  TRUE
    SKIP
```

Fig. 2.13 - Construção IF em OCCAM

Alguns processos apresentam um comportamento do tipo multiplexador de canais, isto é, estão atentos a um conjunto de canais de entrada, dando atenção ao primeiro que se apresente como estando pronto a comunicar. A construção de *Input* Alternativo, ALT, resolve esta situação e a sua sintaxe encontra-se representada na figura 2.14. Cada *input* pode ser antecedido de uma expressão Booleana que permite ou inibe a comunicação através do canal respectivo, dando-se o nome de guarda ao conjunto. Depois de iniciada, a construção ALT fica em estado de espera até que um canal não inibido se encontre pronto a comunicar. Nesse momento, a mensagem será lida para o *buffer* respectivo, dando-se início ao processo correspondente (tipicamente para processar a informação recém-chegada) e a construção terminará. Um ciclo envolvendo a construção permite um atendimento permanente dos vários canais. A figura 2.15 mostra alguns aspectos práticos de utilização da construção ALT.

```
ALT
  <cond_0> & canal_0 ? buffer0
    <processo_0>
  <cond_1> & canal_1 ? buffer1
    <processo_1>
  <cond_2> & canal_2 ? buffer2
    <processo_2>
  ...
```

Fig. 2.14 - Construção ALT em OCCAM

Se vários canais se encontram na situação de prontos a comunicar, a construção não garante qualquer prioridade no atendimento. Para o efeito utiliza-se a variante PRI ALT que atribui prioridades às guardas pela ordem de escrita, sendo a primeira a de maior prioridade. No exemplo da figura 2.15, o canal_A será atendido sempre que esteja pronto a comunicar. Se canal_A não estiver pronto, a prioridade é passada ao canal_B e assim sucessivamente.

As guardas podem declarar-se só com a componente de *input* (canal_A do exemplo), só com a expressão Booleana (terceira guarda, no exemplo) ou com os dois (canal_B). Uma guarda só com expressão Booleana de valor sempre verdadeiro (*TRUE*) utiliza-se quando não se pretenda que a construção ALT fique em estado de espera por um canal pronto.

```
PRI ALT
  canal_A ? msgA
    P0
  (x >= 0) & canal_B ? msgB
    P1
  (x < 0)  & SKIP
    P2
  ...
```

Fig. 2.15 - Exemplo de variantes da construção ALT em OCCAM

Para replicar um certo número de vezes um processo ou componente de processo nas construções SEQ, PAR, IF ou ALT, utilizam-se replicadores. Na figura 2.16, o replicador da construção SEQ resulta num ciclo convencional, a executar *N* vezes.

```
SEQ i=k FOR N          SEQ
  P(i)                   P(k)
                         P(k+1)
                         P(k+2)
                         P(k+3)
                         ...
                         P(N-1)
```

Fig. 2.16 - Replicação da construção SEQ em OCCAM

Na figura 2.17, o replicador de PAR define o que vulgarmente é designado por Vector de Processos em Paralelo, no caso com N processos numerados de k até N-k-1. A alocação de memória em OCCAM é estática pelo que o número de instâncias N na replicação da construção PAR é um número constante.

```
PAR i=k FOR N          PAR
  P(i)                   P(k)
                         P(k+1)
                         P(k+2)
                         P(k+3)
                         ...
                         P(N-1)
```

Fig. 2.17 - Replicação da construção PAR em OCCAM

Os canais em OCCAM permitem a comunicação, entre dois processos, de informação sob a forma de mensagens. O tipo de informação a transferir através de um canal define-se através de um protocolo e cada canal tem que ser declarado com um protocolo (tal como uma variável se declara com um tipo). Os protocolos podem ser simples se se constituem de um item de tipo simples, sequenciais (vários itens simples em sequência) ou variantes. Neste último caso, cada mensagem é antecedida de um *tag* que indica ao processo receptor qual a variante de informação que está a ser comunicada.

2.6 Soluções Paralelas para *Ray-Tracing*

"*...a reasonable division of tasks between processors in a multiprocessor system is to have one or more dedicated to intersection calculations with ray-generation and shading operations performed by the host.*"

Esta citação é retirada do artigo original de T. Whitted [Whi80] sobre *ray-tracing* e coloca, logo à partida, o desafio da paralelização do algoritmo. Durante a década de 80,

vários foram os trabalhos publicados nesse sentido, explorando diversas vias de paralelização que não só a proposta pelo autor.

Nesta secção, resumem-se os principais métodos de paralelização de *ray-tracing*, segundo um estudo efectuado por S. Green [Gre91], e mencionam-se algumas das implementações por ele referidas e julgadas mais marcantes. A secção termina com a apresentação de algumas implementações mais recentes.

2.6.1 Métodos de Paralelização de *Ray-Tracing*

A discussão dos vários métodos de paralelização do *ray-tracing* e suas variantes leva obrigatoriamente à sua classificação e, para o efeito, várias taxonomias têm sido apresentadas. Por ser bastante exaustiva, apresenta-se na figura 2.18 a taxonomia de Stuart Green [Gre91].

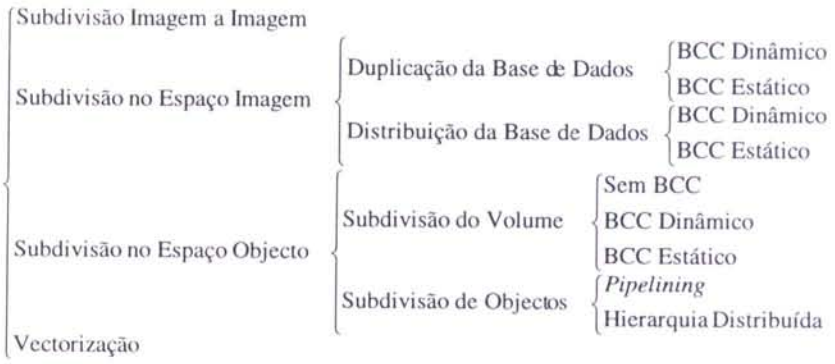


Fig. 2.18 Taxonomia de S. Green para os métodos de paralelização de *ray-tracing*

Esta taxonomia classifica os vários métodos de paralelização de *ray-tracing* atendendo simultaneamente a aspectos mais relacionados com o algoritmo e a outros mais relacionados com técnicas de programação paralela. Descrições de várias implementações (até 1989) de cada um dos tipos mencionados na taxonomia podem ser encontradas na mesma referência e, algumas, resumem-se nas secções seguintes.

Subdivisão Imagem a Imagem

Ao nível mais elevado, o *ray-tracing* pode ser paralelizado pelo cálculo simultâneo de várias imagens, integradas numa animação. Sob o ponto de vista de *ray-tracing*, este tipo

de paralelização é discutível, dado que implementa em paralelo um ciclo exterior ao próprio algoritmo. É vulgarmente utilizado numa rede de computadores.

Subdivisão no Espaço Imagem

A um nível imediatamente inferior, a paralelização do cálculo de subáreas do ecrã é uma possibilidade óbvia. Cada subecrã pode ser entendido como uma tarefa a executar por um processador, havendo que juntar as várias imagens obtidas desta forma. A base de dados dos objectos que constituem a cena pode ser duplicada em todos os processadores ou distribuída.

Subdivisão no Espaço Objecto com Subdivisão de Objectos

O ciclo de processamento de um raio, em *ray-tracing*, inclui a determinação da sua intersecção com o objecto mais próximo da sua origem, actualização do efeito visual que possui sobre o *pixel* a que diz respeito e criação de novos raios, reflectido, transmitido e sensores de sombra. É exactamente a determinação de intersecções que consome a maior parte do tempo de CPU, facto que implicitamente T. Whitted dá a perceber na citação anterior.

Desta forma, um método de paralelização possível baseia-se na separação do algoritmo em dois processos, um que cria raios e faz a actualização correspondente da imagem e outro que recebe esses raios e tenta intersectá-los com os objectos em cena, devolvendo o objecto mais próximo para cada um. Dado que não há qualquer dependência entre os vários raios, o segundo processo pode processá-los em paralelo, o que corresponde a uma paralelização com subdivisão de objectos.

Subdivisão no Espaço Objecto com Subdivisão do Volume

A subdivisão do volume envolvente da cena em subvolumes, juntamente com o facto de um raio se propagar de uma forma linear e coerente, propõe uma forma alternativa de paralelização. A cada subvolume é atribuído um processador que é responsável por intersectar um raio com os objectos que inclui. Encontrada a intersecção mais próxima no interior do volume, é feita a actualização do ecrã e geram-se os novos raios para os quais se repete recursivamente o processo. Cada raio não intersectado é passado para o processador de um subvolume imediato, determinado pela direcção e sentido do raio.

Vectorização

Algumas técnicas de vectorização para arquitecturas apropriadas podem também ser definidas. Um conjunto de raios iniciais, por exemplo, pode ser visto como um vector.

Dado que cada um deles dá origem a vários novos raios, a vectorização dos raios secundários coloca mais problemas.

2.6.2 Alguns Trabalhos mais Relevantes

Como exemplos marcantes de implementações paralelas de *ray-tracing*, são apresentados vários trabalhos, de entre os que são considerados por S. Green:

> Subdivisão do Espaço Objecto e Balanceamento Estático da Carga Computacional: Jevans [Jev89].
>
> Subdivisão do Espaço Objecto e Balanceamento Dinâmico da Carga Computacional: Dippé e Swensen [Dip84], Nemoto e Omachi [Nem86] e Kobayashi *et al.* [Kob89].
>
> Subdivisão do Espaço Imagem: Green [Gre89].

As descrições seguintes, referentes a estes trabalhos, são breves e pretendem somente focar os pontos essenciais dos mesmos, de forma a permitir uma percepção das vantagens e desvantagens dos métodos de paralelização que utilizam.

D. Jevans, em [Jev89], apresenta um sistema paralelo de *ray-tracing* que efectua uma subdivisão do espaço objecto em *voxels* que distribui aleatoriamente pelos processadores. Esta estratégia apresenta-se como uma forma de equilibrar estaticamente a carga computacional e por isso é classificado por S. Green como um sistema paralelo de *ray-tracing* com Subdivisão do Espaço Objecto e Balanceamento Estático da Carga Computacional.

No entanto, o sistema possui várias formas de equilibrar dinamicamente a carga computacional, todas envolvendo processos de intersecção associados a *voxels*:

- Migração de Processos, segundo o qual um processador muito sobrecarregado, em comparação com os seus vizinhos, faz deslocar um processo de intersecções para um processador menos activo. O deslocamento de um processo implica o movimento dos dados associados (objectos incluídos no *voxel* associado) e da lista de tarefas respectiva.

- Duplicação de Processos, que acontece quando um processo se encontra muito sobrecarregado em comparação com outros no mesmo processador. A duplicata é enviada para outro processador e um aviso é feito aos processadores interessados, relatando a duplicação efectuada.

- Exterminação de Duplicatas, utilizada por uma duplicata de processo que quer o seu desapareciemnto, por concluir que está a ter uma actividade muito baixa.

Um raio, ao ser criado, é enviado directamente para todos os *voxels* (processos de intersecção) por onde passa. Todos os processos associados com esses *voxels* iniciam os cálculos de intersecção do raio com os seus objectos.

Quando uma intersecção é encontrada por um processo particular, este envia uma mensagem de terminação para todos os processos activados pelo raio que estejam mais distantes da origem do raio do que ele próprio. Desta forma, a utilização dos processadores é optimizada, mas com um custo adicional que corresponde à realização de cálculos redundantes.

M. Dippé e J. Swensen, em [Dip84], propõem um sistema baseado num vector tridimensional de processadores, com memória distribuída. A cada processador é atribuído um conjunto de sub-regiões resultantes da subdivisão do espaço 3D, de forma a que as sub-regiões de um processador sejam vizinhas e que a vizinhança de processadores seja um reflexo da vizinhança das sub-regiões respectivas.

Desta forma, cada raio viaja no espaço entre regiões e, sempre que necessário, entre processadores, até intersectar (eventualmente) um objecto. Dado que as complexidades das sub-regiões são diferentes[1], a distribuição de carga computacional pelos vários processadores resulta desequilibrada, pelo que o sistema inclui um esquema de redistribuição dinâmica da carga computacional baseado na alteração dos limites de cada sub-região. Assim, uma sub-região sobrecarregada diminui o seu volume, passando alguns objectos para as sub-regiões vizinhas.

Dado que as faces (planas, por motivos de eficiência) e os vértices que definem a fronteira de uma sub-região são compartilhadas pelas sub-regiões vizinhas, a alteração de volume das mesmas é uma tarefa delicada. Não é possível, por exemplo, movimentar livremente os vértices de uma sub-região paralelepipédica e manter a consistência das sub-regiões, também paralelepipédicas, que a rodeiam.

Por isso, os autores usam o tetraedro como forma das sub-regiões. Dado que as faces são triangulares, a movimentação de um qualquer vértice mantém a planaridade das mesmas. Em duas dimensões, substituindo os tetraedros por triângulos, é fácil constatar que as

[1] A complexidade de uma sub-região relaciona-se com o número de objectos que inclui e com a complexidade dos cálculos de intersecções de raios com esses objectos.

arestas mantêm a sua linearidade (figura 2.19) e a extrapolação para as três dimensões é directa.

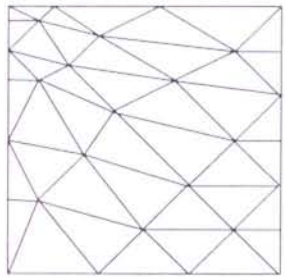


Fig. 2.19 - Ajuste de áreas por movimentação de vértices numa malha triangular

A utilização de tetraedros facilita o equilíbrio de cargas computacionais, mas tem a desvantagem de ser um volume mais custoso em termos de determinação de intersecções com raios e em termos de determinação das faces de entrada e de saída de raios. Note-se que a rapidez desta segunda tarefa é fundamental na passagem eficiente de raios de umas sub-regiões para outras.

K. Nemoto e T. Omachi, em [Nem86], apresentam uma solução semelhante com sub-regiões de forma paralelepipédica (com faces alinhadas com o sistemas de eixos), mas com algumas restrições relativamente aos movimentos possíveis para alteração dos volumes respectivos. Assim, a subdivisão inicial é regular (do tipo da apresentada no algoritmo 3DDDA em [Fuj86]) e, à medida que a correcção da distribuição de carga computacional o exija, cada par de volumes é alterado pelo movimento de uma única face (figura 2.20).

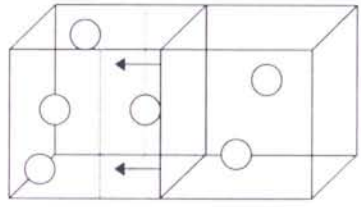


Fig. 2.20 - Ajuste de volumes paralelepipédicos por movimento de uma face comum

Embora mais rápido no que respeita à manipulação directa de sub-regiões, nomeadamente no que se refere à determinação de intersecções, perde, em relação ao

trabalho de Dippé e Swensen, alguma eficiência no que diz respeito ao controlo da distribuição de carga computacional.

H. Kobayashi *et al.*, em [Kob89], propõem um sistema para cálculo de animações por *ray-tracing* que permite redistribuir a carga computacional de uma imagem com base na carga computacional da imagem anterior. Os processadores comunicam entre si através de uma rede em forma de toro e com o processador principal ou hospedeiro (*host computer*) através do barramento do sistema.

Inicialmente, o volume da cena é dividido regularmente em subvolumes que são distribuídos pelos processadores disponíveis, num esquema que os autores designam por alocação distribuída (*distributed allocation*, figura 2.21). Segundo este esquema, a complexidade da cena, no que respeita aos objectos que contém (um objecto pode pertencer a mais do que um subvolume), tende a ser distribuída equitativamente pelos processadores e corresponde a um balanceamento estático da carga computacional para o cálculo da primeira imagem. Cada raio é testado contra os objectos interiores a um subvolume. Não sendo detectada qualquer intersecção, passa a um volume vizinho (de acordo com a direcção do raio) e, eventualmente, a outro processador.

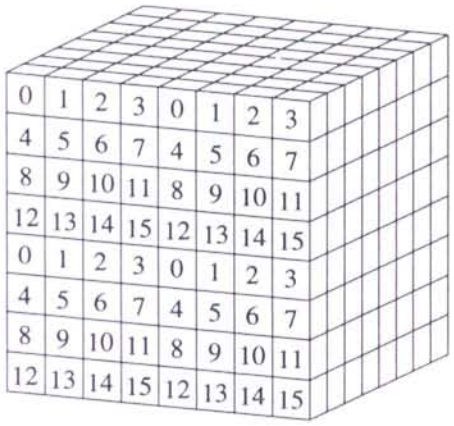


Fig. 2.21 - Alocação distribuída, segundo Kobayashi *et al.*

As acções necessárias ao balanceamento dinâmico da carga computacional efectuam-se entre duas imagens consecutivas, com base na coerência de cargas (*load coherence*): entre duas imagens consecutivas, é pequena a diferença de carga computacional associada a um subvolume. A coerência de cargas permite, ao iniciar uma nova imagem, redistribuir subvolumes pelos processadores com base nas cargas computacionais da imagem anterior.

Após uma imagem ter sido calculada, cada processador envia, para o hospedeiro, uma estatística da carga computacional associada a cada subvolume. O hospedeiro, por sua vez, determina a carga computacional média. Em cada processador, os subvolumes são ordenados por ordem decrescente de carga computacional e os que apresentarem valores inferiores ao valor médio tornam-se candidatos a realocação. Esta é efectuada finalmente pelo hospedeiro e inicia-se o cálculo da imagem seguinte.

Um inconveniente do processo, aliás referido pelos próprios autores, é a perda de vizinhança de subvolumes entre processadores vizinhos quando, entre duas imagens, os subvolumes são redistribuídos. Esta perda implica que um raio, para viajar entre dois subvolumes consecutivos, tenha que viajar através de vários processadores, o que penaliza as comunicações.

S. Green, em [Gre89] e em [Gre89a], apresenta uma versão paralela de *ray-tracing* bastante diferente das anteriores. Enquanto que estas são baseadas na Subdivisão do Volume do Espaço Objecto, a sua solução subdivide o Espaço Imagem, o que oferece muito mais flexibilidade no balanceamento da carga computacional.

No entanto, surge um outro problema, de memória, quando se trate de máquinas do tipo multicomputador.

Nos métodos de subdivisão do espaço objecto já referidos, cada processador necessita somente de conhecer os objectos que se incluem no volume que ele processa. Quando um raio não é intersectado nesse volume, passa ao processador seguinte. Com esta estratégia, é possível distribuir os objectos pelos vários processadores, o que equivale a dizer que a cena deve caber no somatório das memórias individuais de cada processador (memória total). Admitindo um número de processadores relativamente elevado, a memória total tende a ser apreciável, o que permite sintetizar imagens de cenas com alguma complexidade em número de objectos.

Nos métodos de subdivisão do espaço imagem, um raio inicial, lançado através do ecrã para dentro da cena, é entregue a um processador que deve processar toda a árvore de iluminação correspondente a esse raio. Isto pressupõe que o processador tem conhecimento de todos os objectos que fazem parte da cena. Se a cena é replicada por todos os processadores, então a quantidade de memória necessária, num único processador, iguala a capacidade de memória total de um sistema equivalente, mas com subdivisão do espaço objecto.

A solução preconizada por S. Green baseia-se no princípio da coerência de dados inerente ao *ray-tracing*: numa área do ecrã, as árvores de iluminação tendem a ser semelhantes e portanto os objectos intersectados tendem a ser os mesmos. Se cada

processador for responsável por áreas coerentes de ecrã, os acessos à base de dados de objectos tendem a acontecer segundo uma mesma sequência de referências.

Esta situação é análoga à que se encontra em sistemas de gestão de memória virtual em sistemas convencionais, onde a execução de um programa dá origem a uma série de referências a um espaço de endereçamento virtual. Tais referências não acontecem, normalmente, numa sequência aleatória, mas sim de uma forma previsível, devido à localidade de referências no tempo e no espaço (*locality*, [Dei84]).

Baseando-se na teoria da gestão de memória virtual, S. Green desenvolveu o seu sistema paralelo de *ray-tracing* alocando, em cada processador, pequenas quantidades de memória, *caches*, onde tenta manter a descrição dos objectos que tendem a ser referenciados mais vezes. Sempre que um objecto não se encontre na memória local, o processador respectivo envia um pedido para a rede de processadores, para que esse objecto lhe seja fornecido.

O balanceamento de carga computacional é dinâmico. Uma tarefa corresponde a sintetizar a imagem correspondente a uma subdivisão do ecrã. Quando um processador termina uma tarefa, pede à rede de processadores uma nova tarefa. Admitindo que o ecrã se encontra subdividido em pequenas subáreas, esta estratégia resulta numa utilização semelhante de todos os processadores.

2.6.3 Trabalhos mais Recentes

Nesta secção, apresentam-se alguns trabalhos de paralelização de *ray-tracing*, cuja publicação é posterior à elaboração da taxonomia de S. Green [Gre91].

D. Badouel e T. Priol, em [Bad90], [Bad90a] e [Bad94], apresentam dois trabalhos de paralelização de *ray-tracing*, um do tipo Subdivisão do Espaço Imagem com Balanceamento Dinâmico da Carga Computacional e outro do tipo Subdivisão do Espaço 3D com Balanceamento Estático da Carga Computacional.

A solução de subdivisão do espaço imagem é implementada sobre uma arquitectura paralela multicomputador (iPSC/2), e baseia-se num esquema de memória partilhada virtual (VSM-*Virtual Shared Memory*). A memória de dados de cada processador é dividida em dois blocos, um residente e outro destinado a utilização como *cache*. Os blocos residentes de todos os processadores são vistos como um espaço de endereçamento contínuo, organizado em páginas.

Na fase inicial do algoritmo, os vários itens[1] da base de dados da cena são ordenados de acordo com o seu tipo e distribuídos pelos blocos residentes de memória. Posteriormente, o gestor de memória de cada processador faz uso do bloco de *cache*, de forma a reduzir a quantidade de acessos a itens localizados em outros processadores.

Trata-se de um sistema paralelo com decomposição no domínio e orientada às tarefas, correspondendo cada tarefa a uma porção de ecrã. A distribuição de carga computacional é dinâmica, e baseia-se em mensagens de pedido de tarefas que são lançadas para um anel que interliga todos os processadores da rede.

A principais observações prendem-se com o método utilizado para distribuir os objectos pelos blocos residentes de memória.

Em primeiro lugar, os autores afirmam que todos os itens são distribuídos pelos blocos de memória residente. Esta estratégia coloca algumas dificuldades em cenas de maior complexidade.

Em segundo lugar, a distribuição inicial da base de dados pelos blocos residentes de memória não tem em conta os itens mais utilizados por cada processador. Por exemplo, um processador pode ter residente um item ao qual pouco acede e, por outro lado, não ter outro item ao qual acede frequentemente. Obviamente, a *cache* tende a resolver esta situação mas com custos associados.

Também o esquema de paginação da memória em páginas de tamanho fixo não parece ser o mais conveniente dado que, em termos de *cache*, páginas muito utilizadas podem conter itens pouco utilizados.

No segundo trabalho dos autores, do tipo Subdivisão do Espaço 3D com Balanceamento Estático da Carga Computacional[2], atribui-se, a cada processador, uma região paralelepipédica do espaço 3D. A subdivisão do espaço 3D em regiões é feita de forma a reduzir os desequilíbrios da carga computacional e compõe-se das três fases que se seguem:

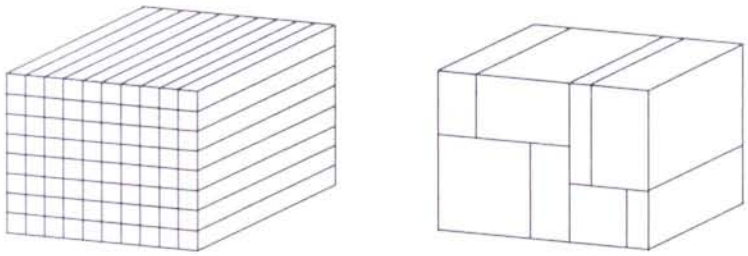
1. Começa por subdividir o espaço 3D em células paralelepipédicas limitadas por planos perpendiculares ao plano *XY*, segundo uma grelha 2D no plano do ecrã, como se mostra na figura 2.22(a).

[1] No artigo original a designação é objectos, o que introduz alguma confusão com o termo objectos da cena 3D.

[2] Os autores referem que este algoritmo é uma evolução do trabalho apresentado em [Pri89] e que é referido na taxonomia de S. Green.

2. Seguidamente, efectua uma subamostragem da imagem, de forma que somente um reduzido número de *pixels* mas distribuídos pela imagem, é calculado. Nesta fase, associa um contador com cada célula de forma a estimar o tempo necessário para processar todos os raios que entram na célula.

3. Finalmente, conhecido o número de processadores P, agrupa as células em P regiões, de acordo com os valores dos contadores e de forma a obter regiões (ou processadores) com idêntica carga computacional. Esta operação é baseada em Partição Binária do Espaço (BSP-*Binary Space Partitioning*) e resulta num número de regiões que é uma potência de dois (bastante conveniente para a arquitectura hiper-cubo utilizada, na qual o número de processadores também é uma potência de dois $P = 2^N$) como se mostra na figura 2.22(b).



(a) (b)

Fig. 2.22 - Balanceamento estático da carga computacional por partição binária do espaço

Cada processador corre um algoritmo *ray-tracing* completo, gerando e processando raios na região que lhe foi atribuída. Quando um raio se desloca para fora da região, o processador envia-o, na forma de mensagem, para o processador correspondente.

Os resultados obtidos com o sistema, medidos na forma de eficiência, não são muito bons. Os autores argumentam que, ao aumentar o número de processadores, aumentam também, por um lado, a quantidade de cálculos necessários à determinação dos pontos de entrada dos raios nas várias regiões (ou processadores) e, por outro lado, o quociente Comunicações/Processamento dado que o tamanho médio das regiões diminui.

V. Isler, em [Isl91], apresenta uma solução semelhante à anterior. A diferença principal está no algoritmo de subdivisão do espaço 3D que reduz não só os desequilíbrios da carga computacional nos vários processadores como também a quantidade de mensagens. Também subdivide o espaço 3D em paralelepípedos limitados por planos perpendiculares

ao plano *XY* e associa um grafo aos subvolumes gerados, no qual os nós representam os subvolumes e os ramos representam as adjacências.

Lança um raio inicial por cada subvolume e processa-o até um determinado nível na árvore de iluminação. Durante esta fase, efectua uma estatística acerca do número de raios processados em cada subvolume e do número de raios que entram e que saem de cada subvolume. Calcula, para cada nó (subvolume), a complexidade dos testes de intersecção $\omega(u)$, baseando-se no número de raios processados e na quantidade de memória necessária para memorizar os objectos aí residentes. Por cada ramo, determina o peso da carga das comunicações respectivas, $d(uv)$.

O grafo é posteriormente dividido em *P clusters* em que *P* é o número de processadores. A movimentação dos limites dos *clusters* (figura 2.23) produz variações em $\omega(u)$ e em $d(uv)$ que são avaliadas e acumuladas em cada *cluster*.

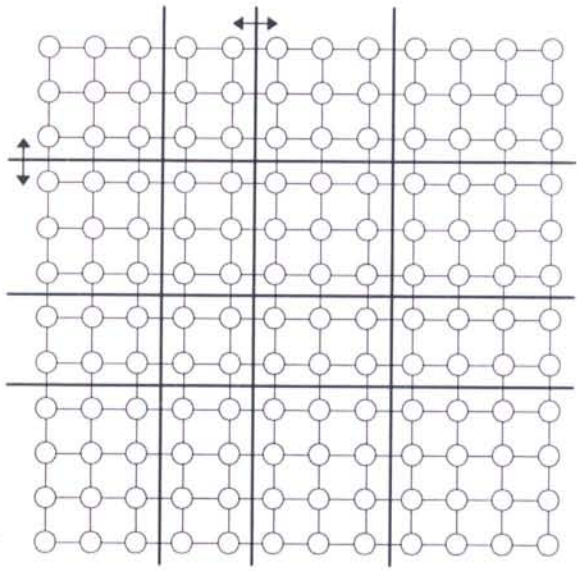


Fig. 2.23 - Grafo inicial de volumes e respectivas adjacências, segundo Isler

O artigo apresenta um algoritmo que efectua a partição do grafo com base nessa avaliação, de forma a distribuir as cargas computacionais pelo vários *clusters*. O resultado obtido é do tipo do que se apresenta na figura 2.24.

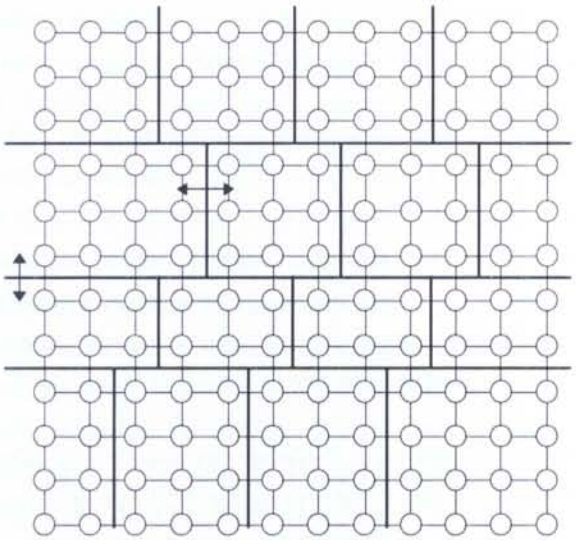


Fig. 2.24 - Grafo inicial de volumes e respectivas adjacências após várias iterações, segundo Isler

Além das observações efectuadas ao método equivalente de D. Badouel *et al.*, mais uma se aplica ao método de Isler: o longo tempo necessário para realizar as operações de partição do grafo (considerado pelo autor como tempo de pré-processamento).

W. Leeffer, em [Lef93], paraleliza o algoritmo *ray-tracing* juntando a decomposição no domínio orientada aos dados e a decomposição no domínio orientada às tarefas [2.2], fazendo-as corresponder a duas tarefas executadas sucessivamente:

T1: Determinar a lista de objectos candidatos à intersecção com um raio numa região (processamento de filtragem). Qualquer processador pode executar a tarefa T1 para qualquer região, pelo que os dados necessários (*filter data*) são replicados pelas memórias locais dos processadores. Corresponde à componente no domínio orientada às tarefas.

T2: Partindo da lista de candidatos, determinar o ponto de intersecção mais próximo e fazer os cálculos de iluminação. Os dados relativos aos objectos (necessários para T2) são distribuídos de acordo com a estratégia de fluência dos raios. Corresponde à componente no domínio orientada aos dados.

Durante a execução de T2 num processador particular, os novos raios criados por reflexão e transmissão, assim como os raios sensores de sombra, são armazenados localmente para posterior processamento. Quando o respectivo *buffer* enche, é enviado para o processador principal (*server*) que o memoriza juntamente com a identificação do processsador de origem.

Um processador subcarregado pede trabalho ao processador principal e este entrega-lhe um *buffer* de raios para aplicar a T1. Os raios, depois de processados, são reenviados para o processador de origem para aplicação na tarefa T2, cuja execução pode originar o envio de vários raios para outros processadores, devido à distribuição dos dados relativos a objectos.

P. Ris e D. Arquès, em [Ris94], retomam as ideias de coerência do espaço objecto apresentadas por S. Green [Gre89] e efectuam uma extensão, criando uma base de conhecimento da relação topológica dos objectos, ou seja, da sua posição relativa no espaço.

O método começa por dividir o ecrã em subáreas a e o espaço 3D em pirâmides $A(a)$ definidas por essas subáreas e o ponto de observação (figura 2.25). Atribui processos de cálculo de *ray-tracing* a cada volume assim definido. As mensagens inter-processos referem-se a informação topológica que permite a cada processo diminuir o número de objectos com os quais deve testar as intersecções de raios.

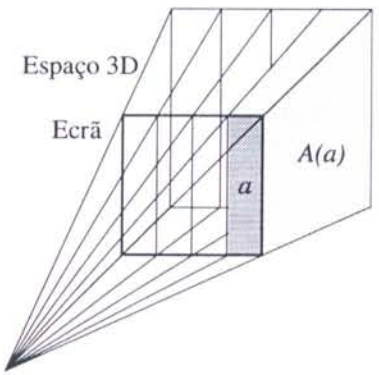


Fig. 2.25 - Áreas ecrã e respectivos volumes segundo Ris e Arquès

Cada processo acede a duas listas de objectos para cálculo de intersecções: uma lista primária a ser usada nos raios iniciais e outra lista para os raios secundários. Quando a caixa envolvente de um objecto é completamente interior a um volume $A(a)$, o objecto pode ser eliminado da lista primária de objectos dos outros volumes.

Cada processo calcula os *pixels* da sua área por linha e coluna (primeira linha e primeira coluna, segunda linha e segunda coluna, etc.). Após o cálculo das primeiras linha e coluna (e recolha de informação dos dois processos opostos à linha e à coluna calculadas), cada processo tem conhecimento dos objectos que intersectam a fronteira da área respectiva.

Com essa informação, um processo efectua um determinado número de deduções acerca da existência ou não de determinados objectos no volume que lhe diz respeito e actualiza a lista primária de objectos de acordo.

Os resultados obtidos pelos autores são encorajadores, mas foram obtidos com uma cena composta por poucos objectos (201 objectos). Dado que o número de mensagens cresce fortemente com o número de objectos em cena, o desempenho global do sistema é fortemente penalizado com cenas mais complexas.

Na taxonomia de S. Green, este trabalho inserir-se-ia no grupo dos que efectuam uma subdivisão no espaço imagem com duplicação da base de dados. Embora não seja explícito quanto ao balanceamento de carga computacional, parece ser do tipo estático.

K. Sun, em [Sun92a], apresenta um sistema a que dá o nome de *Area Sampling Buffer* (ASB) e que faz uso intensivo de *hardware* de *Z-buffer* para amostragem em *ray-tracing*. Seguindo a via do *hardware* dedicado, K. Sun expande o sistema [Sun92], aumentando o seu desempenho pelo recurso à paralelização da componente mais importante do seu sistema: o bloco de amostragem AS (*Area Sampler*). Na figura 2.26 apresenta-se a arquitectura do sistema, designado por ASM (*Area Sampling Machine*).

A base de dados da cena é transformada e depositada no bloco DLM (*Display List Memory*). Este percorre ciclicamente toda a base de dados e coloca periodicamente os descritores dos objectos (polígonos) no barramento DMB (*Display Memory Bus*).

O *software* de *ray-tracing*, do tipo ASB, corre no hospedeiro e gera tarefas para a estrutura CF (*Command-FIFO*) que, por sua vez, configura e inicializa um bloco AS (*Area Sampler*) que esteja disponível. Cada AS é basicamente um *Z-buffer* e efectua, à cadência do DMB, várias operações em *pipeline* com cada objecto: transformação de coordenadas, *clipping*, rasterização e "bufferização". Um bloco AS termina a sua tarefa quando detecta pela segunda vez o primeiro objecto que leu do DMB e envia o resultado da mesma para o bloco OF (*Output-FIFO*) que, por sua vez, o entrega ao bloco ASB.

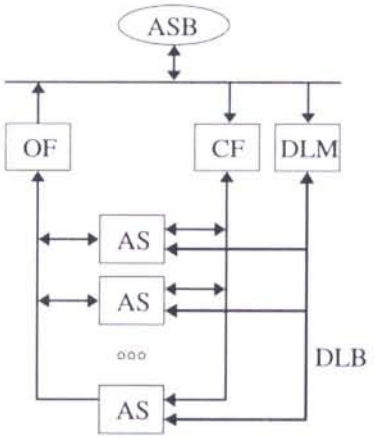


Fig. 2.26 - Arquitectura do sistema ASM, segundo K. Sun

Na taxonomia de S. Green, o sistema ASB inserir-se-ia no grupo de Subdivisão no Espaço Imagem, por paralelizar a amostragem de porções de ecrã. A base de dados é única e é partilhada.

2.7 Resumo

Neste capítulo deu-se uma visão global sobre sistemas de arquitectura paralela, focando assuntos considerados importantes para a boa compreensão das matérias que serão abordadas nos capítulos 4 e 5.

Apresentou-se um resumo com alguns aspectos gerais, desde as arquitecturas, passando pelas técnicas de paralelização, até à definição das grandezas normalmente utilizadas para a medida do desempenho dos sistemas paralelos. (*speedup* e eficiência).

Atendendo a que nos capítulos 4 e 5 serão descritos trabalhos desenvolvidos sobre uma arquitectura baseada em *transputers*, foram focados alguns aspectos relacionados com este tipo de arquitectura. Foram também contempladas as principais construções da linguagem de programação OCCAM, utilizada nesses desenvolvimentos.

Com vista ainda à apresentação desses trabalhos, relacionados com o algoritmo *ray-tracing*, foram descritos os principais métodos de paralelização do mesmo, com base numa taxonomia reconhecida nos meios científicos da especialidade [Gre91]. Efectuou-se uma síntese dos trabalhos de paralelização de *ray-tracing* considerados mais importantes e mencionados na mesma taxonomia. Outros trabalhos, de publicação mais recente, foram também descritos e classificados.

Desta síntese de implementações paralelas de *ray-tracing*, conclui-se que, em termos de arquitecturas para *ray-tracing*, os métodos baseados na subdivisão do volume ou dos objectos requerem estratégias complexas, com custos significativos de *overheads*, para conseguirem bons resultados de balanceamento de carga computacional.

A subdivisão do espaço imagem, nomeadamente em implementação baseada numa arquitectura multicomputador, parece mais interessante do que os métodos anteriores, se bem que apresente alguns problemas como sejam as limitações de memória para o processamento de cenas muito complexas e as estratégias de balanceamento de carga computacional. Para estes, procuram-se soluções que serão analisadas no capítulo 5.

Capítulo 3

Desenvolvimentos em Iluminação Global

3. Desenvolvimentos em Iluminação Global

Por Iluminação Global, em síntese de imagem, entende-se que a iluminação de um ponto depende não só da energia que recebe directamente das fontes de luz, mas também da energia que recebe via reflexão nas superfícies dos objectos que compõem a cena.

São conhecidos como sendo de iluminação global, os algoritmos *Ray-Tracing* e Radiosidade que foram apresentados na secção [1.2], assim como as suas vantagens e desvantagens.

O algoritmo Radiosidade calcula a energia por unidade de área e de tempo, ou Radiosidade, de cada superfície de uma cena. O algoritmo pressupõe que a radiosidade de cada superfície é constante ao longo da sua área e que todas as superfícies são Lambertianas.

O pressuposto da constância da radiosidade ao longo da área de uma superfície obriga à subdivisão desta em *patches*, mas continua a levantar problemas na fase de visualização da cena. Nas condições de visualização com iluminação constante (*flat shading*), a imagem resulta muito grosseira, por apresentar descontinuidades na função de iluminação sobre as fronteiras dos *patches*.

Em alternativa, é vulgar a utilização de métodos de interpolação bilinear para a suavização da imagem. Embora forneça muito melhores resultados do que a iluminação constante, a interpolação bilinear apresenta ainda alguns problemas que interessa corrigir.

A primeira secção deste capítulo aborda o problema de Reconstrução da Função de Iluminação em Radiosidade e apresenta uma solução simples e eficiente baseada em Superfícies Bicúbica.

Por outro lado, o algoritmo radiosidade admite unicamente, por definição, superfícies Lambertianas, o que o torna bastante limitativo. Os espelhos são superfícies especulares puras cuja integração em algoritmos de iluminação global é necessária.

A energia que incide segundo uma dada direcção, numa superfície especular pura, ou espelho, é reflectida segundo uma direcção única, definida por um ângulo de reflexão igual ao ângulo de incidência. Dado que o espelho recebe energia de várias direcções, a função emissão de energia que lhe está associada é complexa e difícil de tratar pelo algoritmo radiosidade, destinado a cenas compostas por superfícies exclusivamente difusas.

Neste caso particular de inclusão de superfícies especulares puras, o algoritmo *ray-tracing* é um potencial a explorar, dado que as reflexões especulares são exactamente a via de transmissão de energia que o algoritmo trata correctamente.

Na segunda secção deste capítulo, apresenta-se uma solução que, por recurso a *ray-tracing*, permite a inclusão de espelhos puros em cenas a sintetizar por radiosidade.

3.1 Reconstrução de Funções de Iluminação em Radiosidade

No mundo real, a energia emitida por unidade de área de uma superfície não é constante. Os motivos podem ser vários, mas os principais são:

- o ângulo de incidência da energia recebida das fontes principais varia ponto a ponto ao longo da superfície;
- outras superfícies opacas projectam sombras e penumbras sobre a superfície.

No algoritmo radiosidade, é suposto que a energia por unidade de área (radiosidade) em cada superfície é constante ao longo de toda a extensão desta, o que é naturalmente contrário à afirmação anterior.

De forma a obter uma simulação bastante aproximada do mundo real é usual, em radiosidade, efectuar-se uma subdivisão das superfícies, em *patches* de menores dimensões. O algoritmo pode assim processar cada *patch* com radiosidade constante e a sequência de vários *patches* com radiosidades diferentes aproxima a variação de radiosidade ao longo da superfície.

Esta solução corresponde a uma discretização da cena e, consequentemente, da função de iluminação respectiva que deve ser reconstruída durante a fase de visualização.

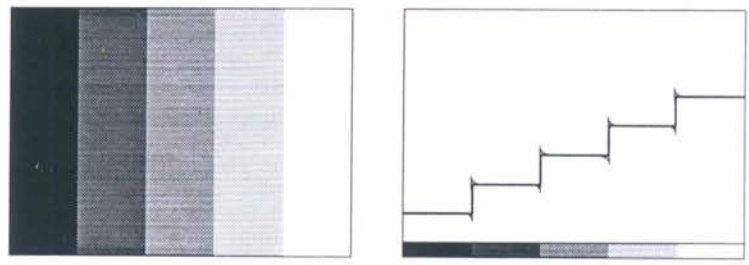
3.1.1 Interpolação Bilinear

Em [Coh85], a reconstrução da função de iluminação é efectuada em dois passos. No primeiro passo e dado que só são conhecidos os valores de radiosidade por *patch*, calculam-se as radiosidades nos vértices, aproximando-as pela média das radiosidades nos *patches* que os compartilham. No segundo passo, uma interpolação bilinear do tipo Gouraud, aplicada a cada *patch*, permite a reconstrução da respectiva função de iluminação, de forma a reduzir descontinuidades na mesma.

A interpolação de Gouraud [Gou71] foi desenvolvida para aplicação em modelos de reflexão locais, para a síntese de imagens contendo objectos limitados por superfícies curvas aproximadas por um pequeno número de faces planas. Um dos principais problemas que lhe está associado é o chamado efeito de *Mach band* [Wat92].

3.1.2 Efeito *Mach Band*

Segundo [Fol90], o efeito *Mach band* foi inicialmente relatado pelo físico austríaco Ernst Mach, em 1865 e, segundo A. Glassner [Gla95], a sua origem ao nível do sistema óptico humano não se encontra ainda completamente explicado. No entanto, o efeito é facilmente observável em casos semelhantes aos das figuras 3.1 e 3.2.

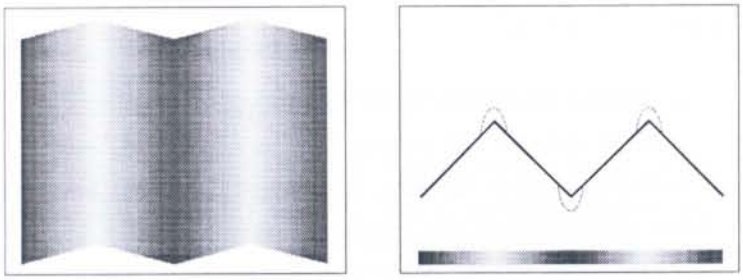


(a) (b)

Fig. 3.1 - Efeito de *Mach band* com iluminação constante

Na figura 3.1, verifica-se que, na vizinhança da fronteira entre dois polígonos de tonalidades de cor diferentes, o polígono mais escuro parece ainda mais escuro e o polígono mais claro parece ainda mais claro.

Na figura 3.2, consegue-se perceber a vizinhança da fronteira entre dois polígonos que partilham uma aresta, apesar da tonalidade de cor ser a mesma, à esquerda e à direita, da linha de fronteira.



(a) (b)

Fig. 3.2 - Efeito de *Mach band* com iluminação interpolada

A interpolação de Gouraud efectua a reconstrução de uma função de iluminação, garantindo uma continuidade de grau C^0 (continuidade na função, descontinuidade na derivada). Assim, subsistem efeitos de *Mach band* do tipo do da figura 3.2, na síntese de imagens por meio do algoritmo radiosidade e a sua eliminação só é possível por meio de interpolações de grau superior ao do método de Gouraud [Bas93].

Apesar da continuidade ser desejável na reconstrução de funções, algumas excepções devem no entanto ser previstas. Descontinuidades nos valores da função (continuidade de grau igual ou inferior a C^{-1}) ocorrem nas regiões de contacto entre superfícies com orientações e/ou características de reflexão/emissão diferentes. Descontinuidades na derivada da função (continuidade de grau C^0) ocorrem nas regiões de fronteira das penumbras.

3.1.3 Outras Interpolações

Nesta secção, resumem-se três métodos de interpolação que garantem uma continuidade igual ou superior a C^1.

D. Kirk e D. Voorhies [Kir90] apresentam um método de interpolação quadrática aplicado a faces de forma triangular, por meio de uma superfície de segunda ordem definida por seis pontos, sendo estes os vértices e os pontos médios dos lados do triângulo.

S. Wong e Z. Cendes [Won87] retomam um trabalho anterior, de Powell e Sabin [Pow77], no âmbito da interpolação por polinómios de segundo grau. Dividem cada

triângulo, de acordo com os triângulos vizinhos, em seis subtriângulos associados a seis *patches* quadráticos de Bézier.

D. Salesin *et al.* [Sal92] aplicam ao algoritmo radiosidade uma solução baseada em interpolação cúbica [Clo65], dividindo cada elemento triangular, em três subtriângulos cúbicos de Bézier. Segundo os autores, a relaxação de restrições dos pontos de controlo (dez pontos de controlo por subtriângulo) permite-lhes definir descontinuidades entre elementos vizinhos.

O método de Kirk e Voorhies tem a vantagem de ser muito simples e eficiente[1], mas pode ainda dar origem a efeito de *Mach band* (especialmente em *patches* de grandes dimensões), por não garantir a continuidade de declive da função de iluminação através da fronteira dos *patches*. Os métodos de Wong e a implementação de Salesin em radiosidade são fiáveis, mas apresentam um grau de complexidade bastante superior, obrigando a um grande número de pontos de amostragem.

O método que se apresenta na secção seguinte foi realizado com o objectivo final de simplificar e tornar eficiente o processo de amostragem, mantendo o objectivo inicial de eliminar o efeito de *Mach band*. Necessita, como entrada, dos valores de radiosidade calculados nos vértices. Dado que estes valores são naturalmente calculados durante o algoritmo radiosidade propriamente dito, nenhuma amostragem extra é portanto necessária.

3.1.4 Interpolação com Superfícies de Hermite

Na pesquisa de outras soluções para o problema, surgem naturalmente os métodos de interpolação bicúbica de superfícies e, dentro destes, as superfícies de Hermite, pela simplicidade da formulação matemática envolvida, com a interpolação da derivada a ser definida pelos planos tangentes à função a interpolar.

Nesta secção, apresenta-se a formulação das superfícies de Hermite adaptada à interpolação dos valores de radiosidade num *patch*. Uma formulação genérica e mais exaustiva pode ser encontrada na bibliografia da especialidade, nomeadamente em [Fol90] e [Far90].

Seja um *patch* rectangular, definido no domínio $s,t \in [0,1]$ e seja $B(s,t)$ a função a interpolar na forma de Hermite:

[1] O método foi desenvolvido para ser implementado em *hardware*.

$$B(s,t) = S \cdot M_H \cdot G_H \cdot M_H^T \cdot T^T \qquad \text{Eq. 3.1}$$

tal que:

$$S = \begin{bmatrix} s^3 & s^2 & s & 1 \end{bmatrix}; \qquad T = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \qquad \text{Eq. 3.2}$$

$$M_H = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad \text{Eq. 3.3}$$

$$G_H = \begin{bmatrix} B_{00} & B_{01} & \frac{\partial B}{\partial t}\Big|_{00} & \frac{\partial B}{\partial t}\Big|_{01} \\ B_{10} & B_{11} & \frac{\partial B}{\partial t}\Big|_{10} & \frac{\partial B}{\partial t}\Big|_{11} \\ \frac{\partial B}{\partial s}\Big|_{00} & \frac{\partial B}{\partial s}\Big|_{01} & \frac{\partial^2 B}{\partial s \partial t}\Big|_{00} & \frac{\partial^2 B}{\partial s \partial t}\Big|_{01} \\ \frac{\partial B}{\partial s}\Big|_{10} & \frac{\partial B}{\partial s}\Big|_{11} & \frac{\partial^2 B}{\partial s \partial t}\Big|_{10} & \frac{\partial^2 B}{\partial s \partial t}\Big|_{11} \end{bmatrix} \qquad \text{Eq. 3.4}$$

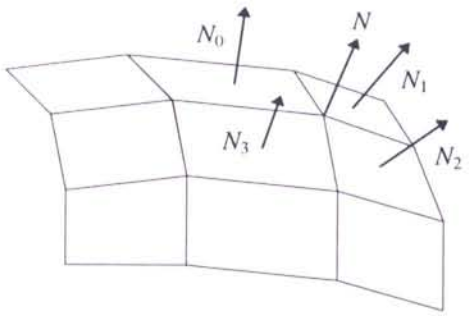
Os parâmetros de entrada na forma superfícies de Hermite (matriz G_H) são as radiosidades ($B_{s,t}$) nos vértices, as derivadas parciais ($\frac{\partial B}{\partial s,t}\Big|_{s,t}$) e as derivadas mistas ($\frac{\partial^2 B}{\partial s \partial t}\Big|_{s,t}$). À partida, somente as radiosidades $B_{s,t}$ são conhecidas, pelo que é necessário encontrar valores para as várias derivadas parciais e mistas.

Entre dois *patches* que partilhem uma aresta, a continuidade de grau C^0 implica igualdade das duas curvas de radiosidade (uma de cada *patch*) que se situam sobre a aresta, ou seja, a radiosidade e as derivadas parciais (na direcção da aresta) para as duas superfícies devem ser iguais ao longo da aresta. A continuidade de grau C^1 implica as mesmas condições, acrescentadas da igualdade de derivadas parciais e mistas cruzando a aresta. Em ambos os casos, as igualdades referidas permitem economia de tempo de

processamento e de espaço em memória, dado que os cálculos de continuidade são efectuados ao nível de vértices e de arestas, sendo estes comuns a dois ou mais *patches*.

3.1.4.1 Planos Tangentes à Função de Iluminação

O cálculo das derivadas necessárias para a definição da matriz de parâmetros G_H passa pela determinação dos planos tangentes à função de iluminação, em cada vértice onde a continuidade seja implícita [Bas93a]. O processo de determinação desses planos, ou das respectivas normais, é semelhante ao processo utilizado em iluminação suavizada (*smooth shading*), na determinação da normal de um vértice: conhecidas as normais das facetas planas que partilham esse vértice, a normal do vértice é aproximada pela média das normais das facetas como se mostra na figura 3.3 [Wat92].



$$N = \frac{N_0 + N_1 + N_2 + N_3}{4}$$

Fig. 3.3 - Cálculo aproximado de uma normal num vértice, pela média aritmética de quatro componentes extraídas das facetas que o partilham

Seja o *patch* representado na figura 3.4, definido pelos vértices V_0-V_3 e sejam as respectivas radiosidades, já conhecidas, B_0-B_3, definidos no sistema de coordenadas *stB*.

Defina-se, no sistema de eixos *stB*, o plano auxiliar que passa pelo ponto central $B_0(V_0)$ e pelos pontos auxiliares $B_1(V_1)$ e $B_3(V_3)$ e determine-se a respectiva normal N_i. Este plano corresponde, no sistema de eixos *stB*, a uma das facetas da figura 3.3, assim como a normal N_i corresponde a uma das normais N_1-N_4. Efectuando a mesma operação com os outros *patches* que compartilham o vértice central V_0, é possível aproximar a normal à função de iluminação naquele vértice pela média aritmética das normais N_i calculadas. A normal à função de iluminação determina o plano tangente à mesma no vértice V_0 que, por sua vez, contém informação necessária para o cálculo das derivadas necessárias à definição da matriz G_H. A função de reconstrução nos *patches* que compartilham o vértice central possui assim, na vizinhança do vértice, continuidade de grau C^1.

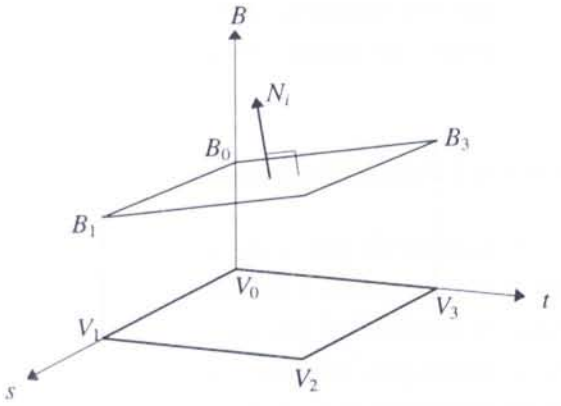


Fig. 3.4 - As radiosidades B_0, B_1 e B_3 permitem determinar uma componente da normal à função de iluminação no vértice V_0

Na figura 3.5 representa-se, em 2D, o processo mencionado para a obtenção dos planos tangentes nos vértices V_2 e V_3, conhecidas as radiosidades B_1, B_2, B_3 e B_4, respectivamente, nos vértices V_1, V_2, V_3 e V_4.

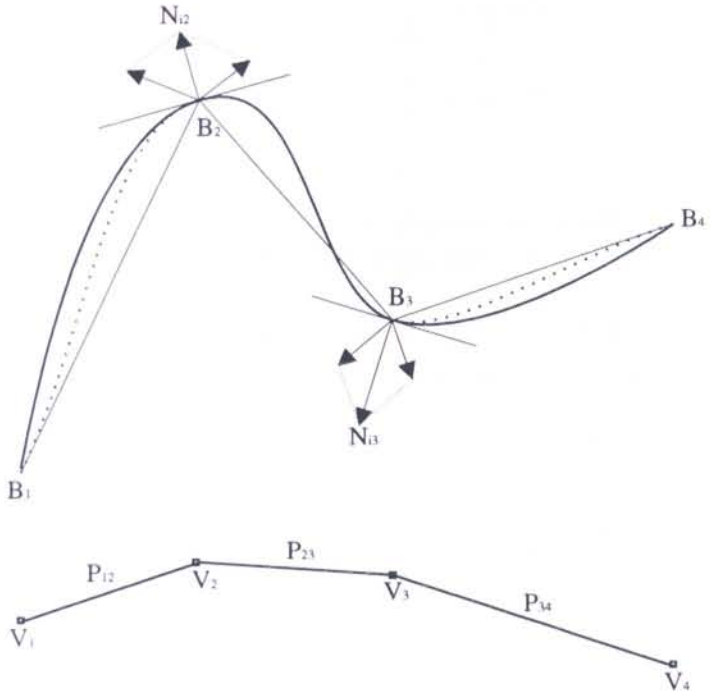


Fig. 3.5 - A normal à função de iluminação, calculada pela soma das normais calculadas em *patches* vizinhos (visão bidimensional)

Considerando V_2 como vértice central, por exemplo, o segmento $\overline{B_1 B_2}$ representa o plano auxiliar no *patch* P_{12} definido a partir das radiosidades nos vértices respectivos

$B_2(V_2)$ e $B_1(V_1)$; o segmento $\overline{B_2B_3}$ representa, de forma semelhante, o plano auxiliar no *patch* P_{23}. A normal N_{i2} à função de iluminação é obtida pela média aritmética das normais aos dois planos auxiliares e contém informação sobre o plano tangente à função de iluminação no vértice V_2. A função de iluminação (linha curva a cheio, na figura), uma vez sujeita ao declive do plano tangente, possui, garantidamente, continuidade de valor e de primeira derivada (C^1).

Para a formalização do cálculo das derivadas parciais e mistas a utilizar na definição da matriz G_H, considere-se a figura 3.6 onde se representa a variação da função de iluminação segundo uma direcção abstracta d (que pode ser s, t ou $s + t$), a partir do vértice central V. O vector N_{id} é a projecção da normal N_i sobre o plano definido por N_g e d.

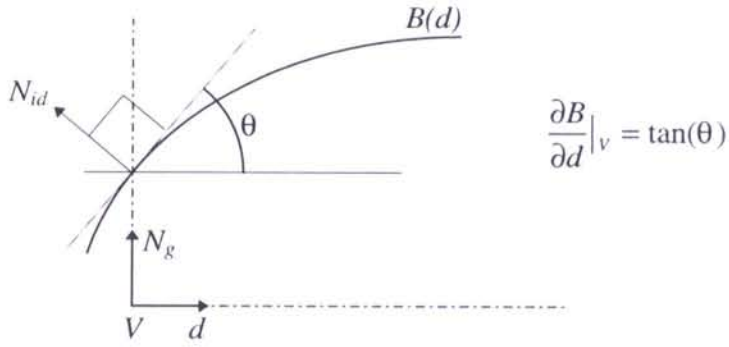


Fig. 3.6 - O plano tangente à função de iluminação e o cálculo de uma derivada parcial

Da figura 3.6 e atendendo à perpendicularidade de N_{id} com o plano tangente e de N_g com d, facilmente se conclui que:

$$\frac{\partial B}{\partial d}\Big|_V = \frac{\sin(\theta)}{\cos(\theta)} = \pm\frac{|N_g \times N_{id}|}{N_g \cdot N_{id}} \qquad \text{Eq. 3.5}$$

O sinal da expressão deve ser escolhido de acordo com o sentido do respectivo vector d. Se os vários *patches* que compartilham o vértice em causa V são complanares, não é difícil encontrar o sinal com base no cosseno do ângulo formado pelos vectores d e N_{id}.

Quando os objectos em cena são limitados por superfícies curvas aproximadas por facetas planas, não existe necessariamente complanaridade de *patches* vizinhos e a determinação do sinal tem de ser efectuada de outra forma. Refira-se que, embora os *patches* não sejam complanares, partilham a normal geométrica N_g e, por se pretender

que apresentem continuidade de grau C^1 no vértice V, partilham a normal à função de iluminação N_i. Um exemplo de uma destas situações é apresentado na figura 3.7, onde N_g e N_i são comuns aos dois *patches* P_1 e P_2.

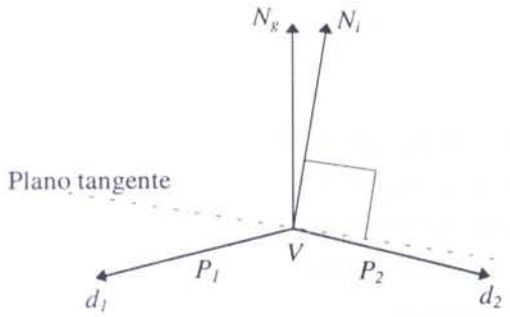


Fig. 3.7 - Dois *patches* vizinhos não complanares

Seja, na figura 3.7, o plano α que contém o vector N_g e que é perpendicular ao plano definido pelos vectores N_g e N_i; considerem-se os dois hemiespaços β_1 e β_2 que α define respectivamente à sua esquerda e à sua direita.

De acordo com as posições relativas do vector N_g e do plano tangente, a função de iluminação na figura 3.7 é decrescente da esquerda para a direita, quaisquer que sejam as direcções de d_1 e de d_2. No entanto, dado que o declive se pretende definido em relação aos sentidos de d_1 e de d_2, diremos que é crescente no hemiespaço β_1 e decrescente no hemiespaço β_2, o que coloca o problema da determinação do sinal na base da classificação dos dois hemiespaços.

O declive é positivo no hemiespaço $\beta_{d>0}$ limitado pelo plano α e definido por um vector n, perpendicular a α, tal que:

$$n = \left(N_i \times N_g\right) \times N_g \qquad \text{Eq. 3.6}$$

Se um qualquer vector d se insere no hemiespaço $\beta_{d>0}$ (facto que pode ser determinado pelo cosseno do ângulo formado pelos dois vectores n e d), então o declive da função de iluminação segundo d é positivo. A equação 3.5 fica:

$$\frac{\partial B}{\partial d}\Big|_v = \frac{n \cdot d}{|n \cdot d|} \cdot \frac{|N_g \times N_{id}|}{N_g \cdot N_{id}} \qquad \text{Eq. 3.7}$$

A Eq. 3.7 permite determinar, da matriz G_H, as derivadas parciais em ordem a s e a t, assim como as derivadas mistas, substituindo d respectivamente pelos vectores s, t e $s+t$.

Em grande parte das situações, cada vértice é completamente rodeado por vários *patches* apresentando continuidade da função de iluminação e, nesta situação, o método descrito possui informação suficiente para produzir uma reconstrução bastante aproximada da função de iluminação.

Em situações de fronteira, no entanto, os *patches* que apresentam continuidade da função de iluminação rodeiam só parcialmente o vértice em causa. A menor quantidade de informação disponível provoca desvios na função de iluminação reconstruída, como acontece nos vértices V_1 e V_4 da figura 3.5, onde a função calculada toma a forma representada a ponteado, exigindo-se por conseguinte uma correcção.

Dado que as radiosidades nos vértices são conhecidas, a informação em falta é sempre do tipo declive no vértice de fronteira (tomando como exemplo o *patch* P_{12} na figura 3.5, a informação em falta é o declive no vértice V_1). Com a informação disponível ou seja, radiosidade no vértice V_1 e radiosidade e declive no vértice V_2, é possível definir uma parábola de cuja equação se pode extrair um boa aproximação do declive em falta, completando a informação necessária para a definição da matriz G_H.

3.1.4.2 Implementação

A técnica de interpolação bicúbica para a reconstrução de funções de iluminação em radiosidade encontra-se implementada como extensão a um pacote de software de radiosidade de domínio público, da autoria de S. Pattanaik e funciona a três passos: *meshing*, cálculo da função de iluminação nos vértices e *rendering*.

Durante a fase de *meshing*, as superfícies são divididas em *patches*, gerando-se os respectivos vértices e arestas. A partilha de vértices e arestas por *patches* diferentes é indicador, para as fases seguintes, de que é desejável a continuidade da função de iluminação. As arestas e vértices onde a função de iluminação é descontínua, duplicam-se, provocando a separação física dos *patches* respectivos. As causas de descontinuidade na função de iluminação são:

- descontinuidade das características de reflectividade das superfícies,
- descontinuidade das características de emissividade das superfícies,

- descontinuidade da normal geométrica,
- fronteira entre uma área de penumbra e uma área iluminada ou em sombra[1].

A fase de cálculo da função de iluminação utiliza um algoritmo radiosidade que, por *shooting*, determina a radiosidade em cada vértice e, posteriormente, os planos tangentes à função de iluminação e as derivadas parciais e mistas.

Durante a fase de *rendering*, determinam-se o *patch* geométrico e as respectivas coordenadas paramétricas (s,t) de cada *pixel*. Conhecidos que são os correspondentes *patches* de reconstrução (um por cada componente de cor), desenvolve-se a interpolação bicúbica das radiosidades, utilizando a equação 3.1, para cada um.

3.1.4.3 Resultados Obtidos

Na figura 3.8(b) apresenta-se uma imagem de teste calculada pelo algoritmo radiosidade segundo o método descrito de reconstrução da função de iluminação. Corresponde a uma sala, com uma única fonte de luz que é uma janela na parede esquerda. Em frente à janela, existe um polígono suspenso que projecta sombra e penumbra na parede da direita. A discretização das superfícies em *patches* é propositadamente grosseira (cerca de 140 *patches*) com o objectivo de fazer salientar o efeito de *Mach band* (a parede em frente é discretizada numa matriz de 3x3 *patches*).

[1] A detecção automática, *à priori*, desta situação não está resolvida.

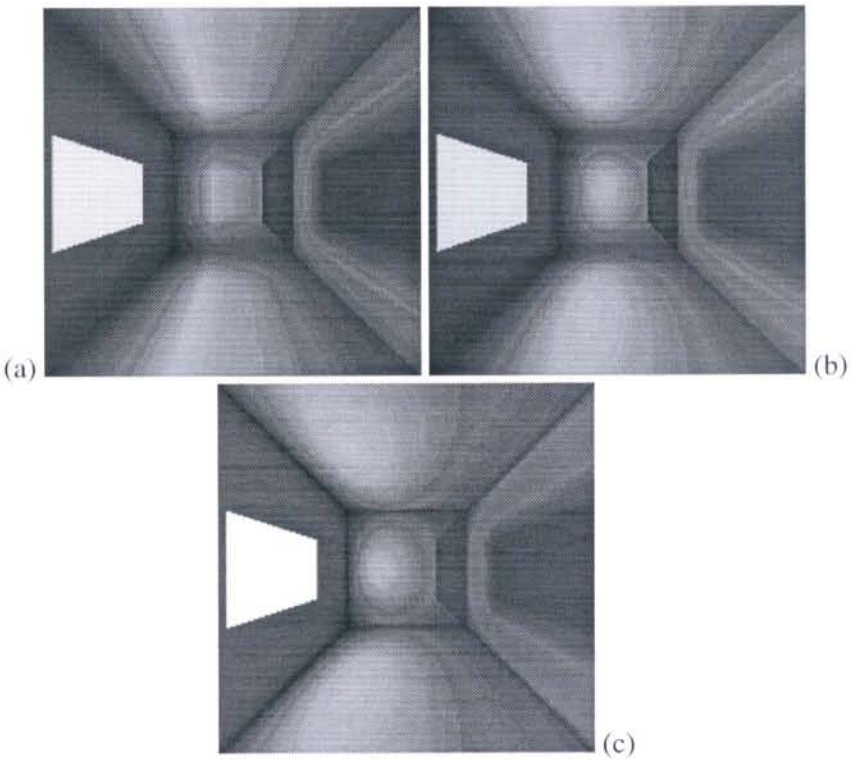


Fig. 3.8 (original na Secção de Cores) - Imagens calculadas com interpolação bilinear (a) e bicúbica (b) e imagem de referência (c)

A imagem de referência, para efeitos de teste, foi gerada com uma discretização da cena muito mais fina (cerca de 14000 *patches*) e apresenta-se na figura 3.8(c). De forma a avaliar visualmente a melhoria de qualidade da imagem introduzida pelo método, a imagem de teste foi ainda comparada com a da figura 3.8(a) que possui uma discretização semelhante, mas que é resultante de uma função de reconstrução do tipo interpolação de Gouraud.

Comparando as figuras 3.8(a) e 3.8(b), é notório o desaparecimento do efeito de *Mach band* na parede em frente, no tecto e no chão. Visualmente, as figuras 3.8(b) e 3.8(c) são bastante semelhantes, apesar da diferença acentuada na granularidade da discretização das superfícies.

Para uma medida mais correcta das diferenças entre as três imagens, apresenta-se na figura 3.9 um gráfico que representa o nível de coloração (do canal vermelho) de uma linha de varrimento situada sensivelmente a meio das imagens. De forma a facilitar a sua interpretação, incluem-se comentários sobre as zonas da imagem a que correspondem os segmentos principais do gráfico.

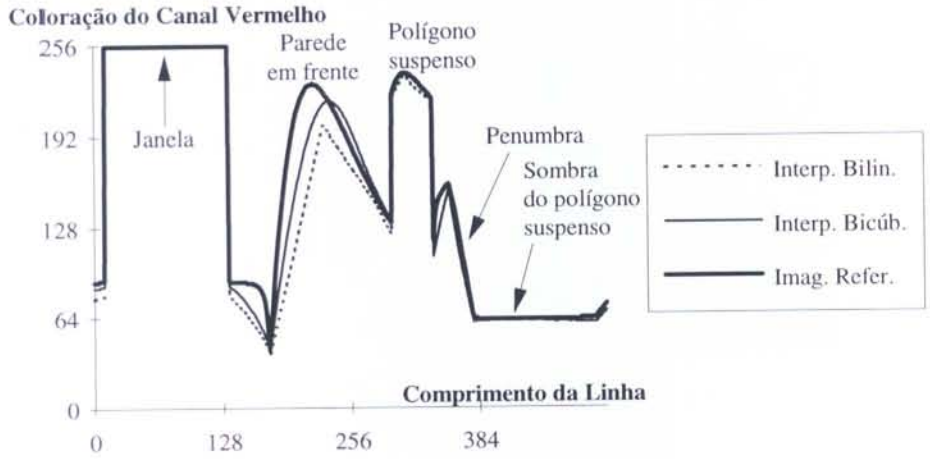


Fig. 3.9 - Análise gráfica da linha de varrimento central

Como se pode observar, a curva correspondente à interpolação bicúbica, é mais próxima da curva da imagem de referência do que a curva correspondente à interpolação bilinear. O principal problema que se nota na interpolação bicúbica é o desfasamento que se encontra entre os dois valores máximos na zona da parede em frente, sendo no entanto os valores de pico relativamente aproximados (valores de 229 e de 216, a que corresponde um erro relativo de 5.7%).

Na zona de penumbra produzida pelo polígono suspenso, as fronteiras dos *patches* foram previamente marcadas como sendo elementos de descontinuidade. Nesta zona e para a figura 3.8(b), foi utilizada a técnica da equação da parábola para aproximar os declives em falta, garantindo-se no entanto a continuidade de grau C^0 da função de iluminação. A mesma técnica foi utilizada nas fronteiras das paredes e do polígono suspenso mas sem garantir qualquer continuidade. Aqui, o erro tende a ser um pouco superior, embora sempre inferior ao que se verifica com a interpolação bilinear.

3.2 Inclusão de Superfícies Especulares em Radiosidade

O algoritmo radiosidade assenta no facto de que, numa superfície, a energia recebida de todas as direcções, ainda que discretas, é acumulada, sendo finalmente reflectida igualmente em todas as direcções. Em radiosidade, não existe portanto a noção da direccionalidade de reflexão que é implícita nas superfícies especulares.

Com o objectivo de incluir superfícies especulares puras na síntese de imagens pelo algoritmo radiosidade e na sequência de trabalhos anteriores, desenvolveu-se um

algoritmo que combina, em dois passos, um algoritmo radiosidade com um algoritmo *ray-tracing*. Por questões de coerência com aqueles trabalhos, a componente de radiosidade do algoritmo é do tipo *shooting*, com cálculo de factores de forma de *patch* para vértice.

3.2.1 Métodos Conhecidos

O método de Wallace *et al.* [Wal87] é um método a dois passos, responsáveis por efectuar os cálculos inerentes a efeitos respectivamente, independentes e dependentes do ponto de visão.

Durante o primeiro passo, um método do tipo radiosidade é aplicado, tendo em conta que a energia que atinge um *patch* difuso receptor é proveniente não só de outros *patches* difusos mas também de reflexões em espelhos. Para o efeito, os factores de forma são calculados, pelo método do hemicubo, tendo em conta os percursos adicionais, pelo que o método necessita de calcular uma imagem virtual da cena por cada espelho (figura 3.10).

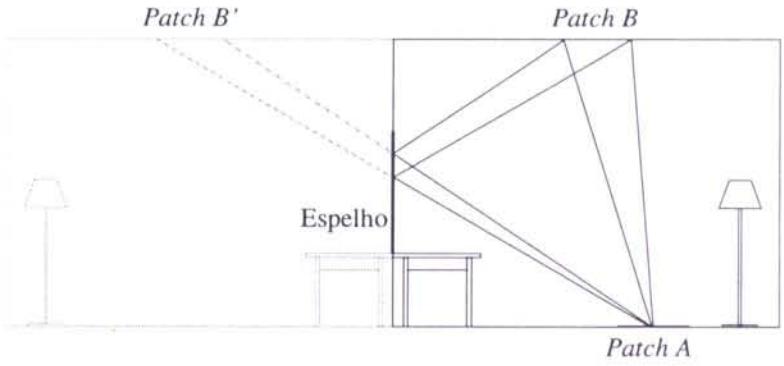


Fig. 3.10 - O *patch A* recebe energia do *patch B* e da sua imagem no espelho

No segundo passo, lançam-se raios do ponto de observação, através do ecrã. Por cada intersecção encontrada, avalia-se o efeito da reflexão especular no ponto de intersecção, amostrando um pequeno conjunto de direcções[1] em volta do raio reflectido; gera-se um novo raio, segundo a direcção óptima de reflexão, que é processado recursivamente, da mesma forma.

[1] Esta amostragem é efectuada por um algoritmo baseado em *z-buffer*, o que permite uma eficiência razoável devido à utilização de *hardware* especializado.

Como principal problema do método, encontra-se a necessidade de determinação de imagens virtuais em espelhos. Com vários espelhos em cena, os percursos adicionais de energia são múltiplos e a complexidade aumenta fortemente.

Immel *et al.* [Imm86] propõem um método que associa, a cada *patch*, um cubo global (em substituição do hemicubo), discretizado em *pixels*. A distribuição direccional da energia em cada *patch* é assim aproximada por um conjunto discreto de ângulos sólidos finitos, definidos pelo centro e pelos *pixels* do cubo global.

A estratégia do cubo global permite conhecer rapidamente, para cada *patch* e segundo uma direcção discreta definida por um dos seus *pixels*, qual o *patch* que é visível. Associando, a cada direcção discreta, uma função de distribuição de energia (por exemplo do tipo BRDF-*Bidireccional Reflection Distribution Function*, [Gla95]), é possível estabelecer um sistema de equações lineares cuja solução permite conhecer a energia que abandona uma superfície numa dada direcção.

A dimensão da matriz que caracteriza o sistema de equações é o principal problema do método. Mesmo se bastante esparsa (o que acontece vulgarmente), leva a quantidades de memória e a tempos de cálculo impraticáveis em termos práticos.

Outros problemas do método relacionam-se com a discretização direccional e surgem quando as superfícies se caracterizam por funções BRDF fortemente direccionais como são os espelhos. Nestes casos é necessário aumentar a frequência de amostragem em volta de cada *patch* ou, o que é o mesmo, aumentar a resolução dos cubos globais, aumentando ainda mais os problemas de armazenamento e de tempo de cálculo.

A proposta de Sillion e Puech [Sil89] reformula o problema no quadro geral da equação de *rendering* [Kaj86], reescrevendo-a com recurso a dois operadores: $\mathcal{J}$, operador reflexão especular e $\mathcal{D}$, operador reflexão difusa. Trata-se de um método a dois passos.

No primeiro passo efectuam-se os cálculos conducentes aos resultados independentes do ponto de visão. Ambos os operadores são usados o que significa que:

- tal como em radiosidade clássica, na determinação de energia recebida por um ponto, são considerados todos os *patches* visíveis desse ponto (operador $\mathcal{D}$);
- para cada um desses *patches*, uma árvore de raios reflectidos e transmitidos deve ser considerada (operador $\mathcal{J}$).

Em consequência, calcula factores de forma estendidos que, embora calculados de outra forma, têm um significado idêntico aos utilizados por Wallace em [Wal87]: indicam qual a percentagem de energia emitida por um *patch* emissor que atinge um receptor, quer directa, quer indirectamente por reflexão em superfícies especulares.

O cálculo dos factores de forma estendidos baseia-se num hemicubo[1], mas enviando raios através das células deste que, recursivamente, se vão reflectindo nos *patches* especulares até atingirem um *patch* difuso. Cada célula do hemicubo fica assim relacionada com o *patch* visível (directa ou indirectamente), segundo a direcção que correspondente à célula.

No segundo passo, um algoritmo do tipo *ray-tracing* simplificado[2] calcula, com recurso ao operador $\mathcal{J}$, os resultados dependentes do ponto de visão.

Este método produz resultados excelentes, embora apresente alguma complexidade na sua implementação. Tem utilização em algoritmos de radiosidade que funcionem na base de *gathering* com factores de forma de *patch* para *patch*.

3.2.2 Factores de Forma com uma Reflexão

J. Wallace, em [Wal89], apresenta um método numérico baseado em *ray-tracing* para o cálculo do factor de forma de um *patch* emissor (área finita) para um vértice (área diferencial) de um *patch* receptor. É possível, como se irá demonstrar, modificar o método de modo a integrar reflexões de energia em espelhos.

O método original de Wallace divide a área do *patch* emissor em N delta áreas; efectua uma amostragem lançando um raio da área diferencial para cada delta área e determinando a respectiva visibilidade δ_i (0 ou 1); determina, finalmente, o factor de forma total pela média aritmética de todos os delta factores de forma, calculados em cada ponto de amostragem, usando a equação:

$$dF_{A_2-dA_1} = dA_1 \cdot \frac{1}{N} \sum_{i=1}^{N} \delta_i \frac{\cos\theta_{1i} . \cos\theta_{2i}}{\pi . r_i^2 + A_2 / N} \qquad \text{Eq. 3.8}$$

onde:

N = número de delta áreas
A_2 = área da superfície emissora
dA_1 = área diferencial no vértice
r = distância entre o centro da delta área e a área diferencial

[1] Os autores apresentam também um método alternativo, mais preciso, que utiliza um plano paralelo ao *patch*, em substituição do hemicubo.
[2] Simplificado porque não lança raios sensores de sombra e porque não efectua cálculos de iluminação, já efectuados durante o primeiro passo.

θ_1 = ângulo formado pela normal à área diferencial e a recta que a une ao centro da delta área

θ_2 = ângulo formado pela normal à delta área e a recta que une o seu centro à área diferencial.

Na dedução desta equação, cada delta área encontra-se aproximada por um disco de igual área. Se as delta áreas forem aproximadamente quadradas, então o erro é pequeno.

Ainda segundo Wallace, a radiosidade num vértice 1 devida à iluminação por um *patch* emissor 2 é assim obtida por (note-se a ausência da área diferencial dA_1):

$$B_1 = \rho_1 . B_2 . A_2 \frac{1}{N} \sum_{i=1}^{N} \delta_i \frac{\cos\theta_{1i} . \cos\theta_{2i}}{\pi . r_i^2 + A_2 / N} \qquad \text{Eq. 3.9}$$

onde:

B_1 = radiosidade no vértice 1 devida ao *patch* emissor 2

B_2 = radiosidade do *patch* emissor

Um estudo do percurso da energia transportada de um *patch* para um vértice com reflexão num espelho, permite reescrever a equação anterior, de forma a calcular a radiosidade num vértice devida à iluminação de um *patch* com reflexão em um ou mais espelhos.

Por simplicidade, a Eq. 3.9 pode ser reescrita:

$$B_1 = \rho_1 . B_2 . \sum_{i=1}^{N} \delta_i \cdot g_i \qquad \text{Eq. 3.10}$$

com

$$g_i = \frac{A_2}{N} \cdot \frac{\cos\theta_{1i} . \cos\theta_{2i}}{\pi . r_i^2 + A_2 / N} \qquad \text{Eq. 3.11}$$

Considere-se a situação representada na figura 3.11, segundo a qual um *patch* receptor R recebe energia de uma delta área E de um *patch* emissor, via reflexão num *patch* especular puro S com dimensões suficientemente grandes para não cortar o percurso de energia.

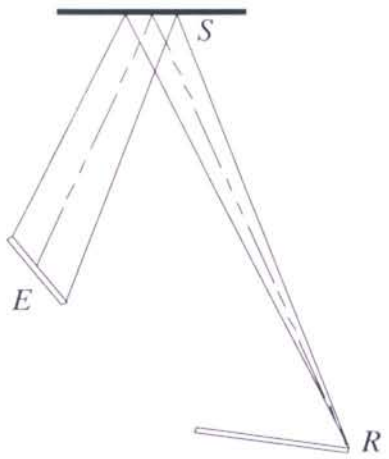


Fig. 3.11 - O percurso de energia desde uma delta área E, até um ponto de um *patch* receptor difuso R, através de um grande espelho

Como forma de facilitar a observação do problema, linearize-se o percurso da energia, substituindo o *patch* R pela sua imagem em S, como consta na figura 3.12 (A_E e A_S são as áreas respectivamente da delta área do emissor e do *patch* especular; θ_E e θ_R são os ângulos formados pela linha central do percurso e pela normal a cada um dos *patches* respectivamente, emissor e receptor).

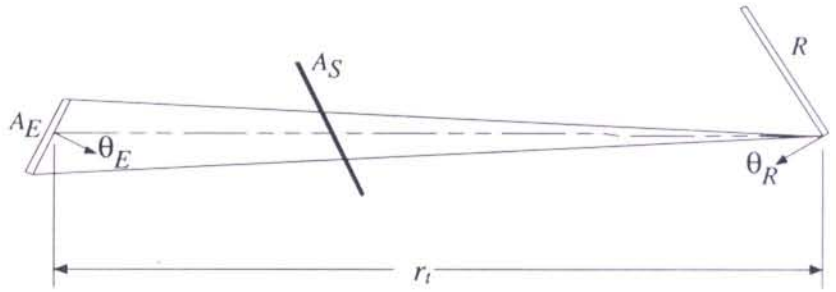


Fig. 3.12 - O mesmo percurso depois de linearizado

Dado que o *patch* especular reflecte toda a energia que recebe, tudo se passa como se os dois *patches* emissor e receptor se vissem directamente, à distância total r_t. A equação 3.11 toma o aspecto:

$$g_i = A_E \cdot \frac{\cos\theta_E . \cos\theta_R}{\pi . r_t^2 + A_E} \qquad \text{Eq. 3.12}$$

o que equivale, na equação 3.9, a utilizar r_t na posição de r, θ_E na posição de θ_{2i} e θ_R na posição de θ_{1i}. Note-se que A_E é a área de uma delta área e portanto $A_E = A_2 / N$.

Admitindo agora que as dimensões do *patch* especular são bastante pequenas, o percurso de energia é cortado, isto é, somente parte da área emissora consegue colocar energia no vértice do *patch* receptor, através do *patch* especular (figura 3.13).

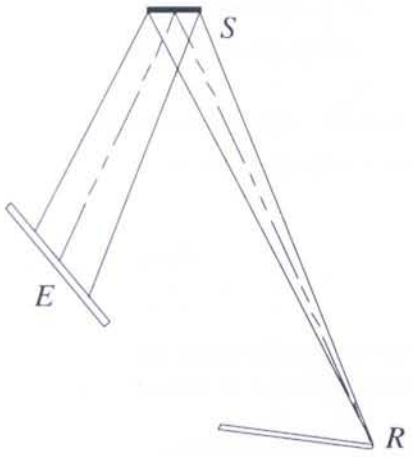


Fig. 3.13 - O percurso de energia desde uma delta área *E*, até um ponto de um *patch* receptor difuso *R*, através de um pequeno *patch* especular

Ao assumir que o *patch S* e os seus vizinhos têm pequenas dimensões, admite-se que, ao linearizar o percurso da energia, os segmentos de recta que unem os centros de *E* e de *S* e o centro de *S* ao vértice de *R* resultam aproximadamente colineares (figura 3.14).

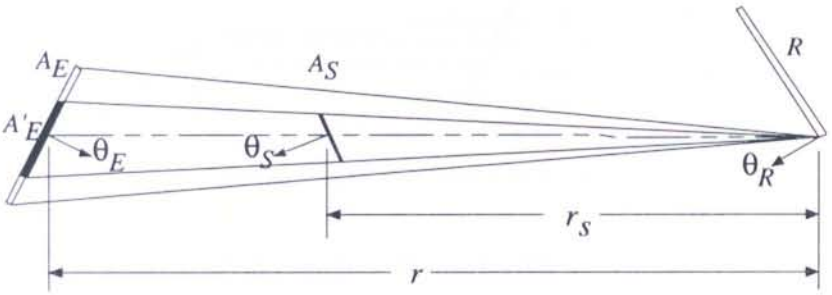


Fig. 3.14 - Geometria do percurso anterior, depois de linearizado

Dado que somente a energia libertada por A'_E atinge o vértice de *R* através de A_S, a equação 3.12 fica:

$$g_i = A'_E \cdot \frac{\cos\theta_E . \cos\theta_R}{\pi . r_i^2 + A'_E} \qquad \text{Eq. 3.13}$$

Considerem-se os seguintes ângulos sólidos ω_S, definido pelo vértice de R e por A_S *e* ω, definido pelo vértice de R e por A_E:

$$\omega_S \approx \frac{A_S.\cos\theta_S}{r_S^2} \approx \frac{A'_E \cos\theta_E}{r_t^2} \; ; \quad \omega \approx \frac{A_E.\cos\theta_E}{r_t^2} \qquad \text{Eq. 3.14}$$

De onde:

$$A'_E = A_E \cdot \frac{\omega_S}{\omega} \qquad \text{Eq. 3.15}$$

Substituindo a Eq. 3.15 na Eq. 3.13 fica:

$$g_i = A_E \cdot \frac{\cos\theta_E.\cos\theta_R}{\pi.r_{ti}^2 + \frac{\omega_S}{\omega} A_E} \cdot \frac{\omega_S}{\omega} \qquad \text{Eq. 3.16}$$

De onde, fazendo $h_i = \omega_S/\omega$ e substituindo na Eq. 3.10, temos a nova equação de radiosidade B_1 no vértice de um *patch* receptor, devida à radiosidade B_2 de um *patch* emissor e reflectida num espelho de pequenas dimensões:

$$B_1 = \rho_1.B_2.A_2 \frac{1}{N} \sum_{i=1}^{N} \delta_i \frac{\cos\theta_{1i}.\cos\theta_{2i}}{\pi.r_{ti}^2 + h_i \cdot A_2 / N} \cdot h_i \qquad \text{Eq. 3.17}$$

Facilmente se constata que esta equação é bastante semelhante a Eq. 3.9. Difere nos termos:

- θ_1 = ângulo formado pela normal à área diferencial no vértice receptor e a recta que a une ao centro do *patch* especular;
- θ_2 = ângulo formado pela normal à delta área emissora e a recta que une o seu centro ao centro do *patch* especular;
- r_t = distância total percorrida desde o *patch* emissor, passando pelo *patch* especular até ao vértice do *patch* receptor;
- h = quociente de ângulos sólidos.

No caso de a energia ser reflectida em vários espelhos até atingir um *patch* difuso, a mesma equação pode ser usada, com alguns cuidados, conforme consta na secção seguinte.

3.2.3 Extensão a uma Sequência de Reflexões

Na secção anterior demonstrou-se a equação matemática que permite efectuar a transferência de radiosidade entre duas superfícies difusas (de um *patch* para um vértice), com reflexão num *patch* especular puro. Nesta secção mostra-se como, a partir das deduções efectuadas, se pode aproximar o cálculo a uma sequência de reflexões em vários especulares puros.

Linearize-se mais uma vez o percurso da energia desde uma delta área emissora difusa até um vértice do *patch* receptor difuso, reflectindo-se em vários *patches* especulares puros (figura 3.15). Admita-se novamente que os centros dos *patches* envolvidos são aproximadamente colineares e definam-se ângulos sólidos para cada um dos *patches* emissor (ω) e especulares (ω_{Sk}), com centro no vértice de R.

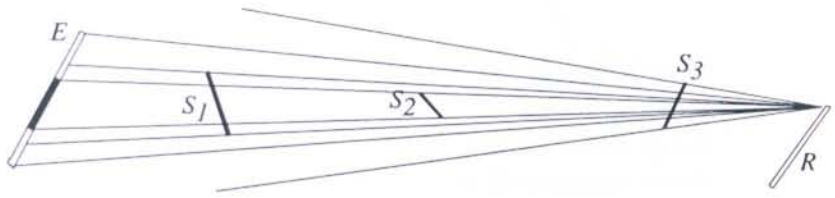


Fig. 3.15 - A transferência de energia de E para R é limitada pelo *patch* especular S_2

A energia libertada pelo *patch* emissor E que consegue atingir o vértice de R é necessariamente limitada pelo menor desses ângulos sólidos. Por exemplo, na figura 3.15, somente a energia reflectida pelo *patch* especular S_2 consegue atingir o vértice R porque define o menor dos ângulos sólidos ω, ω_{S1}, ω_{S2}, ω_{S3}. Portanto, o termo h das equações da secção anterior será:

$$h = \frac{\omega_{min}}{\omega}, \quad \text{com } \omega_{min} = MIN(\omega, \omega_{S1}, \omega_{S2}, \cdots) \qquad \text{Eq. 3.18}$$

Ou seja, a atenuação do ângulo sólido útil corresponde à razão do menor dos ângulos sólidos encontrados pelo ângulo sólido correspondente à delta área emissora. Se os *patches* especulares forem suficientemente grandes, então resulta que $\omega_{min} = \omega$ e, por consequência, a equação 3.17 fica semelhante à equação 3.9 (feitas as devidas ressalvas no que diz respeito aos ângulos θ e à distância r_t).

3.2.4 Implementação

Em [Fer95], apresenta-se uma implementação de um algoritmo iluminação global, funcionando a dois passos e baseado nas deduções anteriores. No primeiro passo, um algoritmo radiosidade calcula a radiosidade de cada vértice, atendendo às formas de transferência energética possíveis. No segundo passo, a imagem é criada por um algoritmo *ray-tracing* simplificado que reflecte raios nas superfícies especulares que encontra e que utiliza, para o cálculo de iluminação em cada *patch*, os valores de energia determinados no primeiro passo.

O primeiro passo é realizado por um algoritmo radiosidade por *shooting*: cada *patch* seleccionado como emissor E distribui energia por todos os outros. Durante este processo, os receptores difusos são processados segundo a regra geral de cálculo de factores de forma por *ray-tracing* e de transferência de radiosidade proposta em [Wal89] (equação 3.9 e mecanismo de transferência de energia Difuso para Difuso, segundo [Wal87]).

Sempre que um *patch* receptor especular S seja encontrado (mecanismo de transferência de energia Difuso para Especular), o raio lançado da delta área emissora é reflectido recursivamente, em outros eventuais *patches* especulares (mecanismo de transferência de energia Especular para Especular), até que um *patch* difuso R seja encontrado (mecanismo de transferência de energia Especular para Difuso).

Conhecido que é o percurso da energia e para cada vértice de R, determinam-se todos os ângulos sólidos ω_S, selecciona-se o menor deles ω_{min} e obtém-se o factor h de modo a possibilitar o cálculo do factor de forma. A radiosidade é finalmente transferida para cada vértice de R por meio da equação 3.17[1].

Dado que o mesmo vértice R pertence tipicamente a dois ou mais *patches*, a aplicação directa da equação anterior resulta em demasiada radiosidade depositada em cada vértice. Uma correcção simples (embora sujeita a algum erro) pode ser introduzida, dividindo a radiosidade, calculada com base nas equações anteriores, pelo número de *patches* que partilham o vértice em questão.

[1] A dedução de equações apresentada em [Fer95] é diferente da que se apresenta neste documento. A expressão final ali obtida deve ser encarada como uma aproximação à expressão 3.17, com um erro que se pode demonstrar, é limitado.

3.2.5 Resultados Obtidos

Seja a cena 3D composta exclusivamente por superfícies difusas da figura 3.16(a) e acrescente-se-lhe um espelho em toda a extensão da parede do fundo.

Dado que toda a energia entra na sala através da janela colocada na parede da esquerda, é esperado que nas áreas mais próximas do espelho, na parede em frente à janela, no tecto e no chão, seja encontrada uma maior quantidade de energia. Por uma questão de ângulos de incidência e de distâncias (que afectam os factores de forma), as áreas mais afastadas do espelho não deverão apresentar acréscimo importante de energia.

A figura 3.16(b) representa a imagem criada pelo método descrito nas condições anteriores e a figura 3.16(c) representa a diferença entre as duas imagens. Os acréscimos de energia referidos são facilmente observáveis. Note-se ainda que, na parede onde a fonte de energia se situa, o aumento de energia é diminuto e deve-se a reflexões de segunda ordem, isto é, a energia das restantes superfícies que se reflecte no espelho.

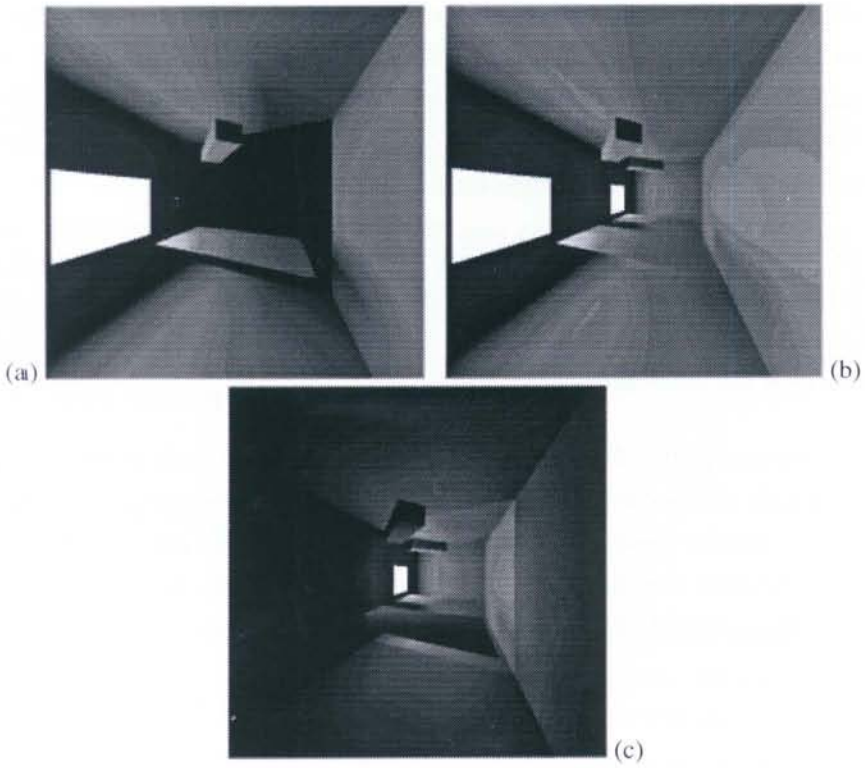


Fig. 3.16 (original na Secção de Cores) - Imagens de uma cena completamente difusa (a), com um espelho (b) e diferença das duas (c)

Considere-se agora uma cena semelhante, na qual se introduz um foco de luz apontado ao chão. O foco tem a forma de um prisma com uma extremidade aberta, possui uma única

superfície emissora localizada na extremidade oposta e as suas superfícies interiores são puramente difusas e de cor clara. Como é esperado e se pode observar na figura 3.18(a), aparece uma mancha luminosa no chão, na zona apontada pelo foco.

Alterando as características das superfícies interiores do foco para puramente especulares, aquela mancha toma um aspecto ligeiramente diferente. Segundo o esquema geométrico da figura 3.17, a zona central da mancha é iluminada directamente pela superfície emissora do foco e, em consequência, é de esperar que a mudança de foco não altere fortemente a iluminação desta zona. Pelo contrário, a orla da mancha é iluminada por reflexão nas superfícies interiores do foco e portanto, quando estas têm características especulares, recebe energia de uma forma mais concentrada.

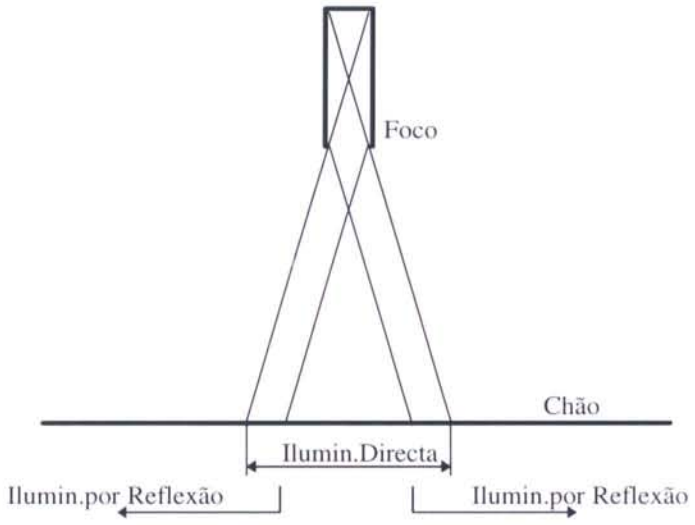


Fig. 3.17 - Geometria da iluminação produzida por um foco de luz

A figura 3.18(b) representa a imagem calculada com um foco de características especulares. As duas imagens são muito semelhantes[1] e, por isso, a imagem diferença apresentada na figura 3.18(c) encontra-se multiplicada pelo factor quatro.

[1] As suferfícies interiores do foco difuso são de côr clara, pelo que também reflectem quase toda a energia que recebem.

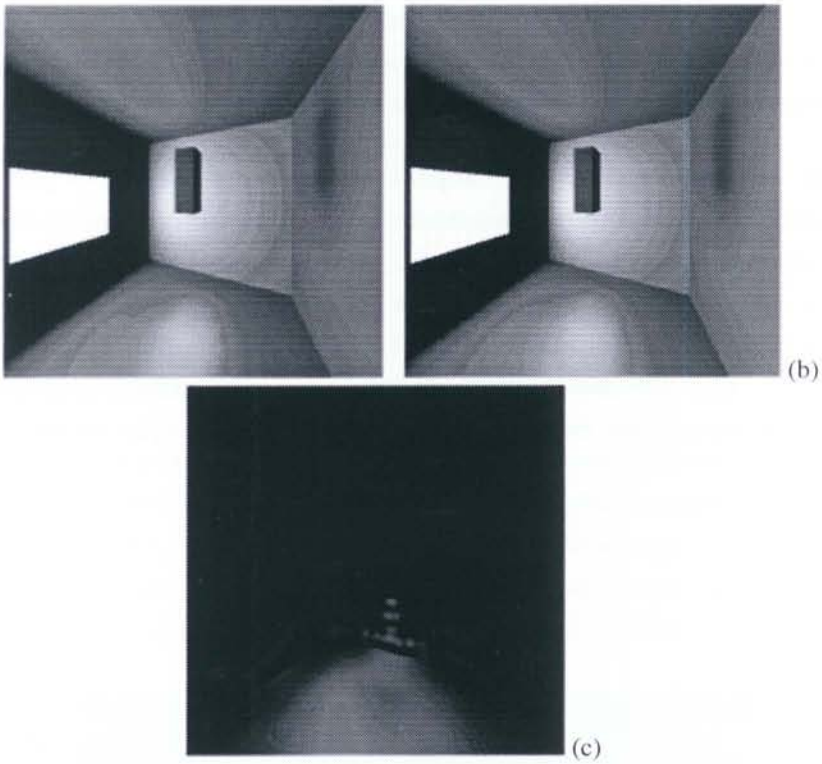


Fig. 3.18 - (original na Secção de Cores) Imagens de uma cena iluminada por um foco difuso (a), um foco especular (b) e diferença escalada das duas (c)

Na imagem diferença pode ver-se, no centro da mancha luminosa, uma zona escura cujo significado é a semelhança entre duas imagens, como é esperado. Na orla da mancha, nota-se uma diferença mais acentuada.

Um dos problemas do método apresentado prende-se com a necessidade de dividir as superfícies especulares em *patches* de pequenas dimensões. Como se pode verificar no exemplo da figura 3.19, se as dimensões dos *patches* especulares são demasiado grandes, um avanço de um *patch* especular em S (durante a fase de *shooting*) provoca um forte desvio do raio reflectido e o *patch* final atingido pode vir a situar-se longe de R. Obviamente, esta situação é causa de um efeito de descontinuidade na distribuição da energia sobre a superfície de R (*aliasing*, na bibliografia). Contudo, em comparação com o método de cálculo de factores forma estendidos por meio de um hemicubo, que apresenta um efeito semelhante [Sil89], este método pode ser vantajoso, dado que a granularidade da divisão pode ser controlada localmente, em cada superfície especular. Além disso, como o aumento da distância dos espelhos para os outros objectos tende a aumentar o efeito referido de descontinuidade, o método torna possível a adopção de granularidades diferentes para os vários espelhos, de acordo com a posição destes

relativamente ao resto da cena. Uma subdivisão adaptativa parece ser uma extensão óbvia para solucionar definitivamente o problema.

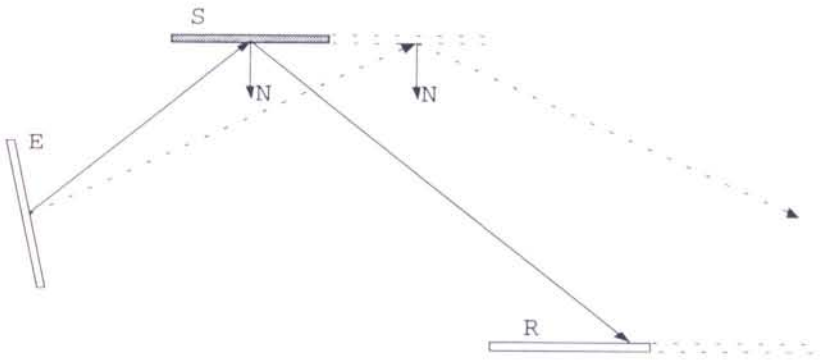


Fig. 3.19 - Descontinidade na distribuição de energia causada por *patches* especulares de grandes dimensões

O aumento consequente do número de *patches* especulares não é, por si só, grande desvantagem, visto que os espelhos reflectem imediatamente toda a energia que recebem. Não acumulando energia, não são nunca seleccionados como *shooters* e portanto não fazem aumentar o número de passos no algoritmo radiosidade.

Também a simplificação de adoptar o menor ângulo sólido numa sequência de reflexões em vários espelhos é algo incoerente quanto à energia libertada pelo *shooter* e recebida pelo *patch* difuso final. No entanto, o menor ângulo sólido é, na maioria das vezes, definido pelo espelho mais próximo do *patch* emissor, pelo que o problema só se verifica em casos pontuais.

As principais vantagens do método são, a sua fácil implementação em algoritmos do tipo radiosidade que funcionem com base em delta factores de forma calculados por *ray-tracing* de *patch* para vértice, e a sua eficiência que resulta do facto de ser necessário lançar um único raio luminoso de uma delta área emissora para um *patch* especular.

3.3 Sumário

O algoritmo radiosidade é um meio excelente de síntese de imagem, baseado no intercâmbio energético entre superfícies puramente difusas e com energia constante ao longo da sua área.

O facto de as superfícies serem supostas com energia constante ao longo da sua área obriga à sua discretização em *patches* de menores dimensões, colocando-se finalmente o problema da reconstrução da função de iluminação ao longo de toda a superfície. O método tradicional de reconstrução, baseado numa interpolação bilinear a partir dos valores conhecidos em cada vértice, garante unicamente continuidade de grau C^0 na função de iluminação sobre as linhas de fronteira dos *patches* da mesma superfície, dando assim origem a efeito de *Mach band*.

Neste capítulo, apresentou-se um método de reconstrução da função de iluminação, baseado numa interpolação bicúbica por meio de superfícies de Hermite. Resumidamente, o método determina a normal à superfície de iluminação em cada vértice, pela média aritmética de normais à superfície de iluminação calculadas nos *patches* que partilham o vértice em questão. As derivadas parciais e mistas necessárias à interpolação de Hermite são deduzidas, em cada vértice, da posição relativa entre as duas normais, à função de iluminação e geométrica.

O método garante a continuidade de grau C^0 da função de iluminação ao longo das linhas de fronteira dos *patches*. Casos há em que essa continuidade não existe, por exemplo quando as linhas de fronteira dos *patches* coincidem com a fronteira dos polígonos a que pertencem. Nestes casos, a informação disponível é insuficiente para uma interpolação bicúbica e, neste contexto, desenvolveu-se um método de cálculo aproximado das derivadas parciais e mistas, baseado numa interpolação de segundo grau.

De acordo com os resultados obtidos, o método elimina o efeito de *Mach band* com um custo computacional baixo. Alguns problemas permanecem, principalmente em *patches* de grandes dimensões que contenham um ponto de energia máxima, cujo posicionamento no *patch* pode não resultar coincidente com o ponto de energia máxima real.

Uma abordagem ao problema de inclusão de superfícies especulares em radiosidade foi também apresentada neste capítulo. O método, aplicável em implementações do algoritmo baseadas em factores de forma de *patch* para vértice, permite a inclusão de espelhos puros, com um custo computacional relacionado com a subdivisão destes em *patches* de pequenas dimensões.

Fundamentalmente, o método utiliza *ray-tracing* para determinar os factores de forma de *patch* para vértice e, nesse contexto, acrescenta os percursos de energia correspondentes à reflexão nos *patches* especulares. Os factores de forma determinados para os percursos adicionais sofrem uma correcção baseada nos ângulos sólidos formados pelos vários *patches* envolvidos e o vértice receptor de energia.

Os problemas que o método exibe relacionam-se com a resolução adoptada para a subdivisão dos espelhos em *patches*. Se a resolução for demasiado grosseira, problemas de descontinuidade na distribuição de energia (*aliasing*) podem surgir.

Capítulo 4

Experiências com *Ray-Tracing* com Nível Crescente de Realismo

4. EXPERIÊNCIAS COM RAY-TRACING COM NÍVEL CRESCENTE DE REALISMO 4.1

4. Experiências com *Ray-Tracing* com Nível Crescente de Realismo

Os sistemas de desenho assistido por computador (CAD-*Computer Aided Design*) em ambientes de produção são, hoje em dia, uma necessidade indiscutível. As leis da concorrência obrigam a uma constante remodelação dos produtos em catálogo assim como ao cumprimento de prazos de entrega muito curtos o que, em conjunto, não se compadece com os métodos tradicionais de projecto manual.

Integrados nesses sistemas, desejam-se recursos de síntese de imagem capazes de criar imagens realistas de novos produtos, com o objectivo de avaliar a sua apresentação visual, evitando tanto quanto possível o fabrico das amostras respectivas, sempre moroso e dispendioso. Mais ainda, a antevisão do efeito visual potencialmente provocado por alterações a efectuar sobre produtos já catalogados é, pelas mesmas razões, uma facilidade desejada.

Tome-se, como exemplo, a execução de um projecto de *design*. Trata-se de uma tarefa faseada segundo o nível de detalhe que o desenhador pretende impor a cada momento [Bon92][Qua92].

As primeiras fases do projecto correspondem à concepção genérica dos objectos e traduzem-se normalmente na elaboração de esboços grosseiros dos mesmos. A quantidade de soluções preconizadas pelo *designer* nestas fases iniciais é tipicamente grande.

Pelo contrário, nas últimas fases do projecto ultimam-se detalhes que podem ser relacionados não só com a forma dos objectos mas também com outros atributos como sejam cores, tipos de materiais, texturas das superfícies, etc. A filtragem de soluções ao longo da execução do projecto possibilita que a quantidade de soluções em causa, nestas fases, seja já bastante baixo.

Os vários níveis de solução associados às diferentes quantidades de soluções a testar têm implicações na algoritmia a utilizar para a síntese de imagens nas múltiplas fases de execução do projecto.

Nas primeiras fases, a modelação dos objectos pode ser grosseira pelo que não se torna necessária uma síntese de imagens de grande qualidade. Por outro lado, a grande quantidade de imagens a sintetizar implica a utilização de métodos de síntese bastante rápidos. Em conclusão, a síntese de imagens baseada em algoritmos de cálculo de visibilidade e iluminação local é perfeitamente adaptada nas primeiras fases de um projecto de *design*.

Nas últimas fases, os objectos encontram-se modelados na sua forma quase final, bastante detalhada, com atributos de aspecto que podem ser bastante complexos. É vulgar ainda a necessidade de integração dos objectos projectados num meio ambiente adequado[1] o que, necessariamente, aumenta a complexidade da imagem e introduz novos efeitos de iluminação, não contabilizados pelos modelos de iluminação local. Nas fases finais de um projecto de *design* é necessário um algoritmo de síntese de imagem capaz de produzir imagens de grande nível de realismo. Com alguma cautela, pode ainda afirmar-se que as restrições de rapidez de cálculo nestas fases são menores. Um algoritmo *ray-tracing* é, pela qualidade das imagens que produz, bastante indicado para as fases finais de um projecto de *design*.

A satisfação de necessidades computacionais de síntese de imagem com base em algoritmos variados, ainda que em fases diferentes de um projecto, pode ser difícil. Por um lado, a passagem brusca de um nível de realismo muito baixo a um nível de realismo substancialmente mais elevado, não raras vezes conduz a resultados inesperados e desagradáveis. Por outro lado, a coexistência de implementações diferentes, muitas vezes recorrendo a *hardware* especializado, acaba por tornar complexa a arquitectura do sistema utilizado, encarecendo-a necessariamente.

Sob o ponto de vista de utilização, o recurso a um único algoritmo de síntese de imagem é portanto a solução mais adequada, desde que cumpra os requisitos correspondentes às várias fases de um projecto.

[1] A produção de imagens para fins publicitários é uma necessidade destas fases.

Os tempos de processamento são o principal óbice à utilização assídua de algoritmos de síntese de imagens, principalmente os mais realistas. O algoritmo *ray-tracing*, como consta na secção [1.2.1], produz imagens de boa qualidade e oferece grandes facilidades ao nível da variedade de primitivas gráficas que admite. Em contrapartida, consome largas horas de tempo de processamento para produzir a imagem de uma cena medianamente complexa. Para as primeiras fases de um projecto, em que as restrições de tempo são maiores, não é portanto defensável a síntese de imagem por *ray-tracing*.

Além disso, o algoritmo *ray-tracing* é, sob o ponto de vista de geração de uma imagem, intrinsecamente sequencial. A imagem é calculada *pixel* a *pixel*, seguindo um trajecto correspondente à sucessão das linhas de varrimento do ecrã, o que acentua ainda mais o problema dos tempos de processamento. Efectivamente, o operador tem que esperar pelos momentos finais de cálculo de uma imagem para ter a possibilidade de a avaliar, eventualmente para a rejeitar, o que não torna prática a sua utilização, mesmo nas fases finais de um projecto.

Um algoritmo baseado em *ray-tracing*, mas com a capacidade de deixar o utilizador prever com bastante antecedência o que virá a ser o resultado final da imagem, é pois uma solução adequada, dado que permite a sua utilização em todas as fases do projecto. O termo *Ray-Tracing* com Nível Crescente de Realismo corresponde a um mesmo algoritmo com a capacidade de gerar imagens cuja qualidade evolui no tempo, permitindo portanto a visualização precoce de aproximaçãoes da imagem final.

Neste capítulo, descrevem-se duas implementações experimentais de *ray-tracing* com nível crescente de realismo. Uma, sobre uma arquitectura tradicional uniprocessador, obtida a partir de um algoritmo *ray-tracing* tradicional desenvolvido localmente [Cos89]. A outra, mais próxima da solução que se pretende final, é desenvolvida sobre uma arquitectura paralela baseada em *transputers*.

4.1 Justificação

Na área dos algoritmos de visibilidade, são conhecidos dois trabalhos com idênticos propósitos, ou seja, capazes de produzir imagens com qualidade progressiva no tempo.

F. Jansen e J. Wijk, em [Jan83], apresentam um problema semelhante que se propõem resolver com base em métodos de previsão de imagens antes do cálculo da imagem final.

Os métodos de previsão evoluem desde uma representação por linhas até uma representação por poliedros com cálculo de visibilidade, aplicação de texturas e cálculo de iluminação local suavizada pelo método de Gouraud [Gou71].

A imagem final, se as previsões forem de acordo com as pretensões do operador, é gerada por um algoritmo de visibilidade mais evoluído. Os objectos em cena tomam então uma representação baseada em *B-spline patches* e, por subdivisão destes até à dimensão do *pixel*, os cálculos são efectuados, um *pixel* de cada vez.

L. Bergman *et al.*, em [Ber86], vão mais longe na análise teórica do problema, comparando, num gráfico que se reproduz na figura 4.1, a qualidade da imagem disponibilizada por várias técnicas de síntese de imagem.

Dois problemas são responsáveis pelo aspecto em degraus que o gráfico apresenta: para uma dada técnica, a imagem não é útil durante os primeiros estágios, quando somente um pequeno número de polígonos ou de linhas de varrimento estão disponíveis; uma vez terminado o processamento de uma técnica, a imagem não aumenta (obviamente) de qualidade.

A solução ideal deve, segundo os autores, corresponder a uma curva do tipo da que é representada a traço forte no mesmo gráfico, ser capaz de decidir qual o trabalho mínimo necessário em cada fase antes de passar para a seguinte, não desperdiçar (tanto quanto possível) resultados anteriores e optimizar a utilização do processador, no sentido de que a qualidade de uma imagem é sempre passível de melhoramentos.

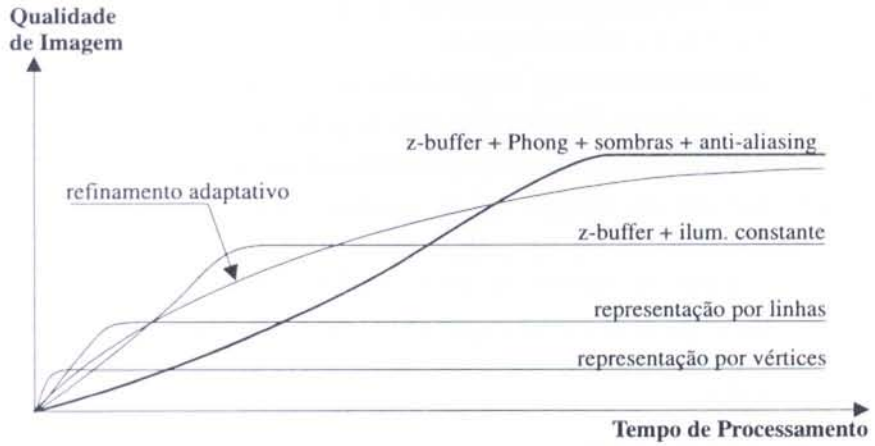


Fig. 4.1 - Evolução da qualidade de imagem por métodos sucessivos de síntese de imagem, segundo L. Bergman *et al.*

A solução que preconizam e que designam por Sistema De Síntese de Imagem Adaptativo baseia-se numa sucessão de sete passos, correspondendo cada um deles a uma técnica de síntese com resultados de qualidade superior ao anterior:

1. Afixação de Vértices com Efeito de *Depth cueing*

2. Desenho de Arestas Visíveis com Efeito de *Depth cueing*
3. *Z-buffer* com Iluminação Constante
4. Projecção de Polígonos de Sombra (se pré-calculados)
5. Iluminação Suavizada de Gouraud (onde necessário)
6. Iluminação Suavizada de Phong (onde necessário)
7. Redução do Efeito de Escada

A qualquer momento, o utilizador pode interromper o cálculo. Esta possibilidade é usada tipicamente em duas situações:

- Quando, numa imagem particular de uma sequência, o utilizador já obteve informação visual suficiente e deseja passar à seguinte;
- Quando conclui que a imagem em causa não corresponde à imagem pretendida.

A solução de Bergman é bastante mais progressiva do que a solução de F. Jansen. Dada a sequência de técnicas que utiliza, tem a vantagem adicional de possibilitar, numa dada fase, um maior aproveitamento de resultados obtidos em fases anteriores. No entanto, repare-se, a relação custo/realismo cresce à medida que as técnicas mais sofisticadas vão sendo adoptadas [1].

Em ambas as soluções, há que contar ainda com a participação do utilizador. Um *Designer* é, normalmente, uma pessoa treinada no sentido de visualizar mentalmente a imagem de um certo modelo que concebeu, ou pelo menos alguns detalhes que considera mais importantes [Por79].

É devido a este princípio que os resultados obtidos com os primeiros passos da solução de Bergman (principalmente) são de grande utilidade. A informação visual que essas imagens contêm é muito baixa e o desconhecimento completo da cena tornaria a sua interpretação impossível. A sobreposição intuitiva da imagem produzida pelo sistema, ainda que rudimentar, com a imagem mental que o utilizador possui, permite a este decidir acerca da validade, em cada momento, dos resultados que o sistema lhe disponibiliza.

Um algoritmo *Ray-Tracing* com Nível Crescente de Realismo é uma variante do algoritmo tradicional de *ray-tracing* destinada a produzir imagens cuja qualidade visual aumenta com o tempo de processamento, à semelhança do que sucede com a solução de Bergman. O utilizador pode assim, com o mesmo algoritmo, obter imagens de menor nível de realismo (nas primeiras fases de um projecto de *design*), sendo para tal necessário que interrompa o processamento no momento adequado. Ou pode, nas últimas fases de um projecto de *design*, obter imagens de elevado nível de realismo, sendo-lhe

ainda permitido interromper o processamento, num estágio bastante prematuro, caso conclua que as imagens não correspondem às suas pretensões.

Admitindo que, mesmo com uma solução progressiva de *ray-tracing*, os tempos de resposta são demasiado longos para a interactividade pretendida e que as soluções de aceleração do algoritmo via *software* são já relativamente eficientes, a implementação de um tal algoritmo sobre uma arquitectura paralela mostra-se uma via interessante que interessa avaliar.

Nas secções seguintes, descrevem-se os princípios básicos de um algoritmo *ray-tracing* com nível crescente de realismo. Apresenta-se uma versão experimental do algoritmo, uniprocessador, desenvolvida a partir de um pacote de *software* já existente e destinada a avaliar as potencialidades dos princípios enunciados. Apresenta-se uma versão multiprocessador, funcionando sobre uma arquitectura baseada em *transputers* e discutem-se os resultados obtidos, apontando os principais problemas e as respectivas vias de solução.

4.2 *Ray-Tracing* Sequencial com Realismo Crescente

O desenvolvimento de uma versão de *ray-tracing* com nível crescente de realismo sobre uma arquitectura paralela, dotada de características próprias, é uma tarefa longa e árdua pela quantidade e novidade de assuntos a integrar. Algumas experiências sobre uma versão tradicional de *ray-tracing* [Cos89] adaptada para o efeito são portanto justificáveis.

A qualidade das imagens produzidas por *ray-tracing* varia com vários parâmetros dos quais se destacam:

- modelo geométrico da cena;
- informação disponível sobre as superfícies dos objectos;
- riqueza do modelo em termos de fenómenos ópticos;
- nível de profundidade da árvore de iluminação;
- número de pontos de amostragem.

O modelo geométrico da cena é um parâmetro obviamente independente do algoritmo propriamente dito e por este motivo não cabe discuti-lo neste texto.

Como informação sobre as superfícies dos objectos entenda-se informação de texturas. Se bem que a existência de texturas possa aumentar a informação visual transmitida por uma imagem, não é possível simplificar o seu processamento nos passos iniciais do

algoritmo dado que interagem com as outras superfícies (via reflexão ou transmissão de raios).

Os fenómenos ópticos tradicionalmente processados em *ray-tracing* são as reflexões difusa e especular dos raios luminosos provenientes das fontes de luz e a reflexão e transmissão especulares puras dos restantes raios [1.2.1]. Por processar estão fenómenos ópticos como a transmissão com refracção de raios provenientes de fontes luz (ausência de cáusticas) e a reflexão e transmissão difusas de outros raios. Além disso, fenómenos como a decomposição da luz em vários comprimentos de onda, com trajectos diferentes para cada um, não são considerados.

A profundidade da árvore de iluminação tem um efeito preponderante no realismo de uma imagem sintetizada por *ray-tracing* visto que é responsável pela componente de iluminação global (ainda que simplificada) do algoritmo[1]. Dado que, por cada nível acrescentado na árvore de iluminação, o número de raios secundários aumenta, melhor aproximados são os efeitos de iluminação secundária, o que implica uma melhoria na qualidade geral da imagem produzida.

Por outro lado, o número de pontos de amostragem tem a ver com a resolução da imagem e com os métodos utilizados para redução do efeito de escada.

Pensando exclusivamente em algoritmia de *ray-tracing* do tipo Whitted [Whi80], as principais áreas de actuação no sentido de incluir a noção de realismo crescente são o número de pontos de amostragem e a profundidade da árvore de iluminação.

4.2.1 Limitação da Profundidade das Árvores de Iluminação

A profundidade das árvores de iluminação pode ser reduzida para as primeiras aproximações de uma imagem, aumentando depois progressivamente até um nível determinado por controlo adaptativo [Hal83]. Assim, para todos os pontos de amostragem, determina-se a árvore de iluminação respectiva, incluindo os raios para as fontes de luz, até um nível da árvore pré-definido. Simultaneamente, a imagem é actualizada de acordo.

[1] No caso limite de a árvore de iluminação ser reduzida a um nível, correspondente a um raio desde o observador até ao primeiro objecto intersectado, o algoritmo *ray-tracing* reduz-se a um algoritmo de iluminação local, o *ray-casting*.

Uma vez atingido o nível pré-definido em todos os pontos de amostragem, pode passar-se a um nível superior. Para tal, é necessário o conhecimento dos raios reflectidos e transmitidos que são descendentes directos dos últimos raios processados.

Duas estratégias podem ser seguidas para o efeito. A mais económica em termos de tempo de processamento é o armazenamento, para posterior processamento, dos raios (reflectidos ou transmitidos) cujo nível iguala ou supera um nível máximo definido pelo utilizador. A estratégia mais económica em termos de memória assenta na repetição da geração recursiva dos raios correspondentes aos primeiros níveis da árvore de iluminação, até ao nível em processamento. Nesta versão experimental é utilizada a segunda hipótese.

Obviamente, a repetição de cálculos penaliza o tempo de processamento. Para atingir um determinado nível na árvore de iluminação, é necessário recalcular a árvore até ao nível imediatamente anterior. Contudo, definição, os raios sensores de sombra não têm descendentes, pelo que somente os raios reflectidos e transmitidos necessitam de ser recalculados até ao nível anterior. Por exemplo, na figura 4.2, a penalização para passar do nível 2 para o nível 3 corresponde a processar um excesso de 3 raios (a tracejado) sobre os 16 que caracterizam o novo nível (a traço forte).

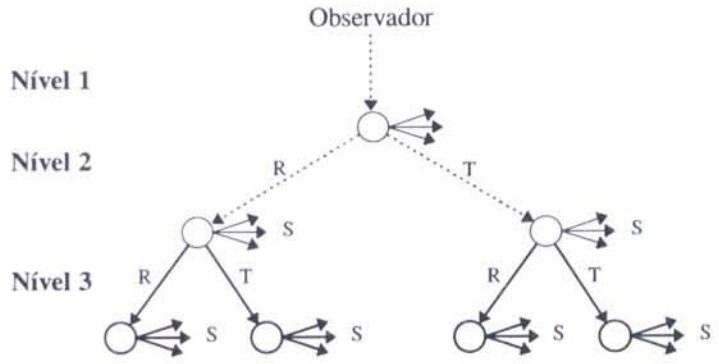


Fig. 4.2 - A penalização, numa árvore completa, para passar do nível 2 para o nível 3 corresponde à repetição da geração de três raios

Em [Lei91] e [Lei93] efectua-se o estudo analítico da penalização introduzida pela passagem do nível m-1 para o nível m na árvore de iluminação, considerando que esta é quase completa (existe um grande número de raios transmitidos através de objectos transparentes). Demonstra-se que, nestas condições e supondo um número N_S de fontes de luz (ou de pontos de amostragem de fontes de luz), a relação entre o número N_A de raios repetidos e o número N_T de raios restritamente necessários para caracterizar o nível m da árvore de iluminação é:

$$\frac{N_A}{N_T} = \frac{1-2^{1-h}}{1+N_S} \qquad \text{Eq. 4.1}$$

Segundo esta equação, a penalização introduzida em cenas com grande número de pontos de amostragem das fontes de luz tende a ser pequena, o que torna o método ainda mais vantajoso em cenas contendo fontes de luz não pontuais, devido à necessidade de amostragem das mesmas. Também em versões de *ray-tracing* com pré-processamento de radiosidade, o grande número de raios que são lançados por cada intersecção favorece o método.

Dado que objectos transparentes existem em menor proporção do que os objectos opacos, a suposição de que a árvore de iluminação é quase completa resulta muitas vezes falsa e invalida a aproximação introduzida na obtenção da equação 4.1. Podem no entanto colocar-se alguns pressupostos adicionais.

Se um raio inicial (a partir do observador) intersecta um objecto opaco então não é gerado qualquer raio transmitido e, se nas intersecções seguintes (níveis seguintes na árvore) suceder o mesmo, então a árvore de iluminação reduz-se a uma lista de raios reflectidos (sucessão de ramos mais à esquerda, na figura 4.2), com um raio por nível. Supondo que é pequeno o número de objectos transparentes e que as suas dimensões são reduzidas, então a maioria dos raios iniciais redundam em árvores deste tipo.

Por outro lado, se o raio inicial intersecta um dos (pressupostamente poucos) objectos transparentes, dá origem a dois novos raios, um reflectido e outro transmitido. O raio reflectido tem grande probabilidade de gerar uma subárvore do tipo da anterior. Quanto ao raio transmitido, uma vez que entra no objecto, acaba por intersectá-lo de novo, agora do lado oposto e gera dois novos raios. O raio transmitido dá origem, a partir deste ponto, a uma subárvore em forma de lista e o raio reflectido intersectará o mesmo objecto num outro ponto e assim sucessivamente. A árvore de iluminação é semicompleta conforme se mostra na figura 4.3 e apresenta uma característica importante: o número de raios a processar por nível é igual ao próprio nível.

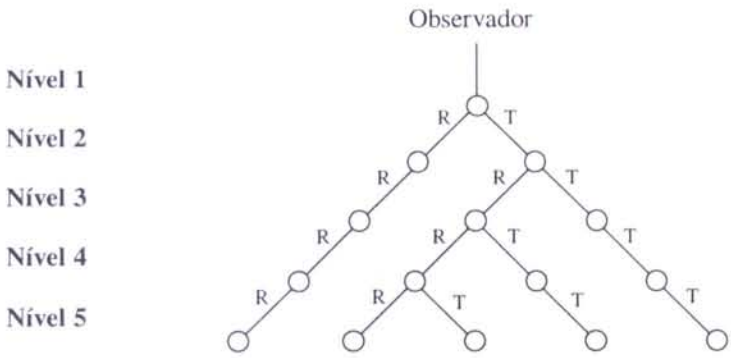


Fig. 4.3 - Uma árvore semicompleta

Uma melhor aproximação do que a anterior (árvores completas em todos os pontos de amostragem) pode ser obtida considerando que cada ponto de amostragem é caracterizado por uma árvore em lista ou por uma árvore semicompleta. Corresponde (aproximadamente) à imagem de uma cena composta por objectos opacos tendo, em primeiro plano, um objecto transparente. Nesta situação, podem definir-se as percentagens de área do ecrã, s e l, ocupadas respectivamente pelo objecto transparente (árvore semicompleta) e pelos restantes objectos (árvore em lista):

$$l + s = 1 \qquad \text{Eq. 4.2}$$

Sejam N_{TS} e N_{AS}, respectivamente, o numero de raios a processar no nível m e o número de raios a repetir para atingir o nível m, numa árvore semicompleta:

$$N_{TS} = m \cdot (1 + N_S) \; ; \qquad N_{AS} = 1 + 2 \ldots m - 1 = \frac{m \cdot (m-1)}{2} \qquad \text{Eq. 4.3}$$

e as equivalentes N_{TL} e N_{AL}, numa árvore em lista:

$$N_{TL} = 1 + N_S \; ; \qquad N_{AL} = m - 1 \qquad \text{Eq. 4.4}$$

Nas condições postas, e considerando a globalidade da imagem, a penalização é:

$$\frac{N_A}{N_T} = \frac{l \cdot N_{AL} + s \cdot N_{AS}}{l \cdot N_{TL} + s \cdot N_{TS}} \qquad \text{Eq. 4.5}$$

e, substituindo recorrendo às equações 4.3 e 4.4:

$$\frac{N_A}{N_T} = \frac{[1+(m/2-1)\cdot s]\cdot(m-1)}{[1+(m-1)\cdot s]\cdot(1+N_S)} \qquad \text{Eq. 4.6}$$

O quadro 4.1 apresenta, para efeitos comparativos, a penalização determinada pela equação 4.6 e a penalização verificada na prática durante o cálculo de uma imagem de teste do tipo das que são utilizadas na secção 4.2.3. A cena inclui 27 fontes de luz pontuais e representa uma sala com uma mesa e uma janela em primeiro plano. Estes dois objectos definem cerca de 13% da imagem caracterizada por pontos de amostragem em árvore semicompleta (s=0.13).

Nível	Determinação Analítica	Verificação Prática
1	0.0%	0.0%
1->2	3.2%	6.2%
2->3	6.0%	10.8%
3->4	8.7%	15.1%
4->5	11.2%	18.6%

Quadro 4.1 - Penalização introduzida pela repetição parcial dos cálculos inerentes às árvores de iluminação

Os valores de penalização determinados analiticamente são ainda visivelmente afastados dos valores verificados para a imagem de teste. Devido às características muito pouco especulares de algumas superfícies (paredes), vários ramos das árvores de iluminação não atingem níveis mais elevados[1], o que faz diminuir N_T e, por conseguinte, aumentar a penalização. Uma maior generalização na determinação analítica da penalização introduzida pelo peso computacional das repetições de raios é difícil pois este, além de depender da quantidade e das dimensões dos objectos transparentes (parcialmente considerados em Eq. 4.6, depende também da sua disposição na cena, com efeitos imprevisíveis sobre os raios secundários (nível 2 e superiores) na árvore de iluminação.

No entanto, contrariamente à equação 4.1, a equação 4.6 denota uma tendência crescente dos valores de penalização com o nível máximo da árvore (tal como os valores verificados na prática) o que torna o método pouco útil para cenas compostas por

[1] O esquema implementado para o controlo da profundidade máxima da árvore de iluminação detecta que os ramos referidos têm uma influência diminuta sobre os correspondentes pontos ecrã e impede a sua propagação.

objectos muito especulares. Admitindo no entanto que o método pode ser selectivamente utilizado só nos primeiros níveis, pensa-se que uma penalização inferior a 20% (obtida, relembre-se, com 27 fontes de luz pontuais) compensa o facto de o utilizador poder prever, com bastante antecedência, a imagem que está a ser calculada.

4.2.2 Limitação do Número de Amostras

A limitação do número de pontos de amostragem de uma imagem complementa o método anterior no sentido de se obterem imagens com realismo crescente em *ray-tracing*. Corresponde ao termo Limitação e Ampliação em Largura mencionada em [Lei91][1] e Controlo da Resolução Espacial mencionada em [Lei93].

Num algoritmo *ray-tracing* tradicional, o número de pontos de amostragem é, no mínimo, igual ao número de *pixels* da imagem a criar. As técnicas utilizadas na redução do efeito de escada podem aumentar fortemente este número.

Numa perspectiva de *ray-tracing* com realismo crescente, as primeiras aproximações de uma imagem podem ser obtidas com um número limitado de pontos de amostragem desde que as condições seguintes se verifiquem simultaneamente:

1. O número de pontos de amostragem possa aumentar com mais tempo de processamento;
2. A amostragem seja distribuída por toda a área interessante da imagem (à partida, todo o ecrã).

Cada amostra definida neste contexto influencia uma certa área circundante que, à medida que o número de amostras aumenta, diminui na mesma proporção. Estas áreas são designadas por Células Imagem.

Desta forma, as primeiras versões de uma imagem em realismo crescente, calculadas com um número pequeno de amostras, são compostas por um conjunto grosseiro de células imagem. Se bem que o aspecto tosco assim obtido torne a imagem imperceptível como um todo, permite ao utilizador, com um tempo adicional de cálculo desprezável, determinar aproximadamente a posição dos objectos assim como alguns efeitos de iluminação. Progressivamente, o aumento do número de amostras faz diminuir a área das células e, em consequência, a imagem vista globalmente tende assintoticamente para a

[1] O termo deriva do facto de, pela diminuição do número de árvores a processar, se diminuir a largura média das mesmas.

versão final. Este processo é semelhante ao proposto em [Jan83] para um algoritmo *ray-casting*.

Cada célula imagem, de formato rectangular, é caracterizada, à partida, pelas amostras tomadas nos seus quatro cantos e pode ser dividida recursivamente em quatro novas células de dimensões inferiores por um processo semelhante ao de amostragem adaptativa utilizado na redução do efeito de escada [Whi80][Cos89].

Em rigor, a decisão de dividir ou não uma célula pode ser delicada, dado que envolve factores como a percepção humana de pequenas diferenças de cor [Tas91] No entanto, num contexto de realismo crescente, pode ser simplificada. O método adoptado, baseado na diferença normalizada entre os valores máximo e mínimo das intensidades encontradas nas amostras da célula, é semelhante ao proposto por D. Mitchell [Mit87] para determinar áreas da imagem cuja variação de sinal obriga a uma super-amostragem. Segundo Mitchell, a resposta não linear do olho humano a variações rápidas de intensidade de luz é aproximadamente modelada pelo contraste:

$$C = \frac{I_{max} - I_{min}}{I_{max} + I_{min}} \qquad \text{Eq. 4.7}$$

Calculado o contraste de uma célula para cada componente de cor, a divisão de uma célula efectua-se se se verificar ter um valor superior a um parâmetro de tolerância, o que permite concentrar o esforço computacional do algoritmo nas áreas da imagem com menor coerência.

No caso da presente implementação de *ray-tracing* com realismo crescente, as várias células imagem não têm necessariamente as mesmas dimensões. Por outro lado, os quatro cantos de uma célula situam-se necessariamente sobre a sua fronteira que por sua vez é comum a células vizinhas. Em conclusão, as amostras disponíveis por célula são, por vezes, em número superior a quatro e situam-se sempre sobre a sua fronteira.

Nestas condições, os valores a utilizar na equação 4.7 são os máximos e mínimos de todas as amostras disponíveis. A afixação de uma célula no ecrã é efectuada por uma interpolação bilinear que utiliza a informação de todas as amostras disponíveis para essa célula.

A subdivisão recursiva e adaptativa de células imagem tal como é descrita nos parágrafos anteriores pode levar à perda de informação visual, isto é, objectos ou efeitos de iluminação, cuja dimensão seja inferior à de uma célula, podem passar despercebidos por não serem detectados por qualquer das amostras efectuadas. T. Akimoto *et al.*, em

[Aki91], propõem um método de aceleração de *ray-tracing*, que designam por PSRT-*Pixel Selected Ray-Tracing*, baseado num método de amostragem semelhante[1]. No entanto e apesar de não perder partes muito finas de objectos (mas longas o suficiente para atravessarem mais do que uma célula), o método não garante que não haja perda de pequenos objectos na sua totalidade.

Seja, no método que se descreve neste texto, uma célula imagem qualquer. Se o contraste determinado é superior ao parâmetro de tolerância, a célula é classificada como não homogénea e é subdividida.

Se, pelo contrário o contraste é inferior àquele parâmetro, então um teste adicional é efectuado de forma a evitar a perda de pequenos objectos. Este teste adicional consiste na verificação da existência de objectos incluídos no volume da pirâmide definida pelo ponto de observação e pela fronteira da célula imagem.

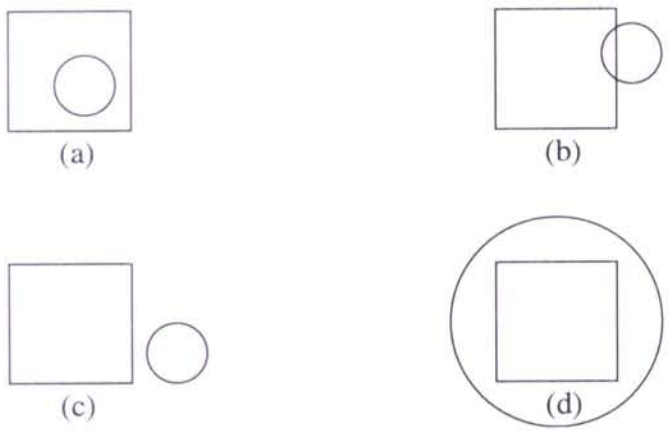


Fig. 4.4 - Teste de inclusão de objectos numa célula imagem

Se como resultado se conclui que não existe qualquer objecto que satisfaça aquelas condições, então a célula é definitivamente classificada como homogénea (casos (c) e (d) na figura 4.4) e é afixada no ecrã. Se existe pelo menos um objecto incluído no volume da pirâmide (casos (a) e (b) na mesma figura), então a célula classifica-se como não homogénea e é necessariamente subdividida, repetindo-se o processo para cada uma das células suas descendentes.

Desta forma, garante-se a existência de pelo menos uma amostra por objecto. Esta amostra, ao diferir das restantes amostras que caracterizam uma célula, provocará a sua

[1] Realmente o método de Akimoto utiliza mais uma amostra por célula, ao centro da mesma, que pode ser efectuada por um de três métodos que descreve e dos quais, dois são aproximados.

subdivisão recursiva que, no limite, termina quando atinge a fronteira da silhueta do objecto em questão. A perda de pequenos efeitos luminosos não é garantidamente evitada por este método.

O teste de inclusão de objectos numa pirâmide é apresentado com pormenor em [Lei91] e [Lei93]. Um teste com uma finalidade semelhante apresenta-se em [Rei95].

4.2.3 Implementação e Avaliação de Resultados

Esta versão experimental de *ray-tracing* com realismo crescente permite ao utilizador controlar, por passos, a evolução da qualidade da imagem por um dos dois métodos descritos: limitação da profundidade na árvore de iluminação e limitação das dimensões das células imagem.

A limitação das dimensões das células imagem efectua-se com base em dois parâmetros definidos pelo utilizador, um de controlo de contraste e outro de limite mínimo de dimensões. Um terceiro parâmetro controla a profundidade máxima da árvore de iluminação e um conjunto de quatro parâmetros adicionais permitem ao utilizador definir uma zona (rectangular) de interesse na qual o algoritmo concentra o esforço computacional.

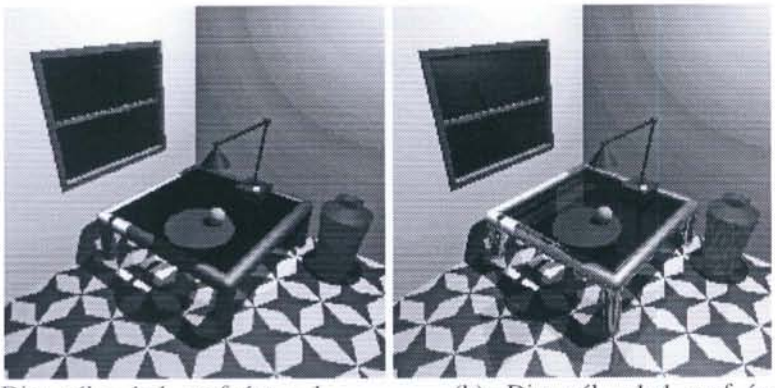
Em cada passo, o utilizador modifica um ou mais parâmetros de forma que a nova versão a calcular da imagem seja de acordo com a(s) via(s) de incremento de qualidade que ele escolhe.

Na figura 4.5 apresenta-se uma sequência de imagens geradas pelo sistema. A imagem (a) corresponde a células imagem de dimensões 1x1 *pixels* e com árvores de iluminação calculadas até ao nível 1. Não são visíveis quaisquer reflexos e todos os objectos em cena parecem ser difusos.

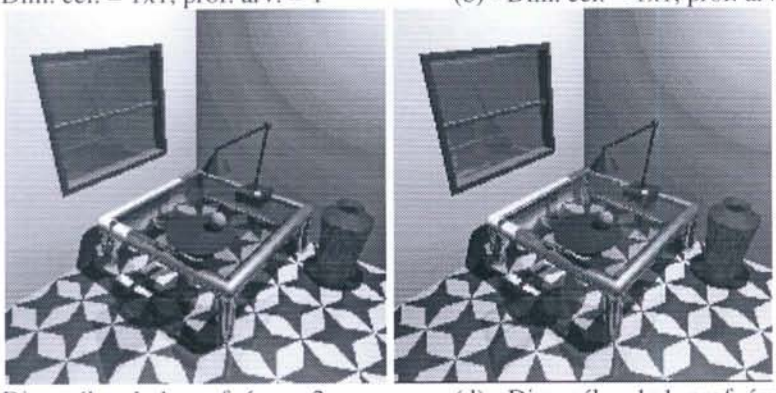
A imagem (b) é obtida a partir da (a), acrescentado um nível nas árvores de iluminação. Começam a notar-se alguns reflexos, nomeadamente na armação metálica da mesa, mas o tampo desta não denuncia qualquer transparência. Efectivamente, os raios gerados por transparência encontram-se no interior do material que constitui o tampo da mesa.

Para a geração da imagem (c), a profundidade das árvores de iluminação passa a ser 3, pelo que são visíveis mais alguns reflexos. Novos raios são gerados por transparência a partir dos raios que se encontravam no interior do tampo da mesa. O chão e as paredes, intersectados por estes novos raios, passam a ser visíveis através do tampo da mesa.

A imagem (d) corresponde a um aumento acentuado nos níveis da árvore de iluminação.



(a) - Dim. cél. = 1x1, prof. árv. = 1 (b) - Dim. cél. = 1x1, prof. árv. = 2

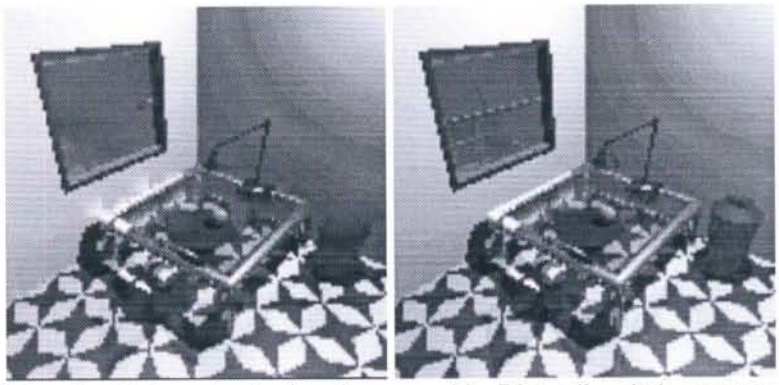


(c) - Dim. cél. = 1x1, prof. árv. = 2 (d) - Dim. cél. = 1x1, prof. árv. = n

Fig. 4.5 (Original na Secção a Cores) - Sequência de imagens obtida por incrementos na profundidade das árvores de iluminação

A sequência da figura 4.6 mostra uma progressão em resolução e em contraste. As imagens (a) e (b) são constituídas por células imagem 4x4 *pixels* e correspondem a árvores de iluminação com nível elevado. A diferença entre as duas é provocada pelo parâmetro de controlo do contraste que está mais relaxado na imagem (a). Devido ao elevado contraste permitido e à interpolação bilinear efectuada no interior das células imagem, notam-se alguns efeitos visuais de alastramento de cor, semelhantes aos que se obtêm por arrasto sobre um desenho a carvão.

A imagem (c) foi obtida a partir da (b), por aumento de resolução (células imagem com 2x2 *pixels*).



(a) - Dim. cél. = 4x4, contraste elevado (b) - Dim. cél. = 4x4, contraste baixo

(c) - Dim. cél. = 2x2, contraste baixo

Fig. 4.6 (Original na Secção a Cores)- Sequência de imagens obtida por progressão em resolução e em contraste das células imagem

4.3 *Ray-tracing* com Realismo Crescente em Arquitectura Paralela

O problema principal da solução descrita na secção anterior [4.2] é a necessidade de recalcular a subárvore de nível *m* para avançar na árvore de iluminação para um nível superior. Este problema reflecte-se em duas formas distintas. Por um lado aumenta o tempo de processamento (embora, como se mostrou, a penalização introduzida não seja demasiado elevada, pelo menos nos casos identificados) e, por outro lado, atribui ao algoritmo um modo de funcionamento por passos o que lhe dá um carácter pouco interactivo. Torna-se pois necessário memorizar raios secundários, no momento da sua geração, para posterior processamento.

Em *ray-tracing* tradicional, de Whitted [Whi80], a intensidade a atribuir a um *pixel* é a intensidade do raio de nível 1 e determina-se pela aplicação da equação 1.3:

$$I_1 = I_a \cdot k_{a,1} + \sum_{S=1}^{N_S} \left[I_S \cdot (k_{d,1} \cdot \cos\theta_{S,1} + k_{r,1} \cdot \cos^{n_1} \phi_{S,1}) \right] + k_{r,1} \cdot I_{r,1} + k_{t,1} \cdot I_{t,1}$$

Eq. 4.8

Nesta equação, $I_{r,1}$ e $I_{t,1}$ são as intensidades dos raios gerados por reflexão e por transmissão a partir do primeiro raio, no ponto de intersecção. Como tal, a equação é recursiva e está de acordo com a visita em profundidade à árvore de iluminação.

Se os raios são armazenados, como se pretende, é garantido que a árvore de iluminação seja visitada exactamente pela mesma ordem. Além disso, por imperativos de memória, é necessário que um raio já processado seja eliminado. Não havendo conhecimento, para um dado raio em processamento, dos seus antecedentes, não existe, à partida, informação suficiente para efectuar a actualização da imagem que lhe corresponde.

Simplifique-se momentaneamente a equação 4.8, por eliminação da parcela correspondente ao raio transmitido (equivale a calcular uma árvore de iluminação reduzida a uma lista de raios reflectidos) e substitua-se $I_{r,1}$ pela intensidade I_2 do raio seu descendente de segundo nível:

$$I_1 = I_a \cdot k_{a,1} + \sum_{S=1}^{N_S} \left[I_S \cdot (k_{d,1} \cdot \cos\theta_{S,1} + k_{r,1} \cdot \cos^{n_1} \phi_{S,1}) \right] + k_{r,1} \cdot I_2 \qquad \text{Eq. 4.9}$$

Da mesma forma, para a intensidade de um raio de nível i na mesma árvore:

$$I_i = I_a \cdot k_{a,i} + \sum_{S=1}^{N_S} \left[I_S \cdot (k_{d,i} \cdot \cos\theta_{S,i} + k_{r,i} \cdot \cos^{n_i} \phi_{S,i}) \right] + k_{r,i} \cdot I_{i+1} \qquad \text{Eq. 4.10}$$

Supondo agora uma sequência de m reflexões a partir do raio inicial, temos que a intensidade a atribuir ao *pixel* é (com $k_{r,0}$=0):

$$\begin{aligned} I_m = & \sum_{i=1}^{m} \left[\prod_{j=0}^{i-1} (k_{r,j}) \cdot I_a \cdot k_{a,i} \right] + \\ & + \sum_{i=1}^{m} \left[\prod_{j=0}^{i-1} (k_{r,j}) \cdot \sum_{S=1}^{N_S} \left[I_S \cdot (k_{d,i} \cdot \cos\theta_{S,i} + k_{r,i} \cdot \cos^{n_i} \phi_{S,i}) \right] \right] + \\ & + \prod_{j=0}^{m} (k_{r,j}) \cdot I_{m+1} \end{aligned} \qquad \text{Eq. 4.11}$$

De onde, fazendo:

$$f_i = \prod_{j=0}^{i-1} \left(k_{r,j} \right); \qquad \text{Eq. 4.12}$$

Se obtém:

$$I_m = \sum_{i=1}^{m} \left[f_i \cdot \left(I_a \cdot k_{a,i} + \sum_{S=1}^{N_S} \left[I_S \cdot (k_{d,i} \cdot \cos\theta_{S,i} + k_{r,i} \cdot \cos^{n_i} \phi_{S,i}) \right] \right) \right] + \left(f_m \cdot k_{r,m} \right) \cdot I_{m+1}$$

Eq. 4.13

Admitindo que, ao nível m, é encontrada uma superfície transparente, a parcela correspondente ao raio transmitido é agora reinserida na equação que fica:

$$I_m = \sum_{i=1}^{m} \left[f_i \cdot \left(I_a \cdot k_{a,i} + \sum_{S=1}^{N_S} \left[I_S \cdot (k_{d,i} \cdot \cos\theta_{S,i} + k_{r,i} \cdot \cos^{n_i} \phi_{S,i}) \right] \right) \right] + \\ + \left(f_m \cdot k_{r,m} \right) \cdot I_{r,m+1} + \left(f_m \cdot k_{t,m} \right) \cdot I_{t,m+1} \qquad \text{Eq. 4.14}$$

Atendendo a Eq. 4.12:

$$f_{r,m+1} = f_m \cdot k_{r,m} \; ; \qquad f_{t,m+1} = f_m \cdot k_{t,m} \qquad \text{Eq. 4.15}$$

tem-se finalmente:

$$I_m = \sum_{i=1}^{m} \left[f_i \cdot \left(I_a \cdot k_{a,i} + \sum_{S=1}^{N_S} \left[I_S \cdot (k_{d,i} \cdot \cos\theta_{S,i} + k_{r,i} \cdot \cos^{n_i} \phi_{S,i}) \right] \right) \right] + \\ + f_{r,m+1} \cdot I_{r,m+1} + f_{t,m+1} \cdot I_{t,m+1} \qquad \text{Eq. 4.16}$$

A interpretação das equações 4.15 e 4.16 é a seguinte:

- numa sucessão de reflexões até à profundidade m da árvore de iluminação, cada raio contribui, para a amostra respectiva, com uma parcela que corresponde à soma da iluminação ambiente, com a iluminação directa das fontes de luz, multiplicada por um factor de influência f;

- sendo unitário o valor do factor de influência de um raio inicial e conhecido o factor de influência *f* de um qualquer raio, os factores de influência dos seus descendentes directos, reflectido e transmitido, determinam-se multiplicando *f* pelos respectivos coeficientes de reflexão e de transmissão k_r e k_t.

Esta interpretação sugere uma alternativa para o algoritmo, mais eficiente em termos de tempo de processamento e mais interactiva (embora bastante mais consumidora de espaço de memória), baseada no armazenamento de raios ainda não intersectados com a cena, para posterior processamento. Para o efeito, é necessário afectar a cada raio, um campo adicional designado por factor de influência (um por cada componente de cor) que permite determinar a percentagem da intensidade de um raio que tem influência sobre a amostra respectiva.

Esta alternativa dá alguma liberdade na ordem pela qual os raios reflectido e transmitido são processados: contrariamente ao *ray-tracing* recursivo, é possível, por exemplo, processar raios de amostras diferentes para um determinado nível e só depois avançar na profundidade da árvore (ou seguir um critério inverso); é possível acrescentar amostras em qualquer momento do cálculo, etc. Esta facilidade de processar raios por qualquer ordem permite a adopção de uma estratégia de processamento selectivo dos raios armazenados, que seja de acordo com a implementação de realismo crescente. Para tal, convém alargar a utilização dos factores de influência aos raios sensores de sombra.

Substituindo, na equação 4.16:

$$f_{S,i} = f_i \cdot (k_{d,i} \cdot \cos\theta_{S,i} + k_{r,i} \cdot \cos^{n_i} \phi_{S,i}) \qquad \text{Eq. 4.17}$$

tem-se:

$$I_m = \sum_{i=1}^{m} \left[f_i \cdot k_{a,i} \cdot I_a + \sum_{S=1}^{N_S} \left(f_{S,i} \cdot I_S \right) \right] + f_{r,m+1} \cdot I_{r,m+1} + f_{t,m+1} \cdot I_{t,m+1} \qquad \text{Eq. 4.18}$$

de onde, partindo de uma amostra vazia (I_0=0):

$$I_m = I_{m-1} + f_m \cdot k_{a,m} \cdot I_a + \sum_{S=1}^{N_S} \left(f_{S,m} \cdot I_S \right) + f_{r,m+1} \cdot I_{r,m+1} + f_{t,m+1} \cdot I_{t,m+1}$$

Eq. 4.19

Em conclusão, os raios sensores de sombra podem também ser processados por uma ordem qualquer, desde que o seu efeito seja afectado por factores de influência apropriados f_S calculados pela equação 4.17.

Mais ainda, a equação 4.19 define um método de cálculo que é incremental, quer no que respeita à definição dos factores de influência, quer mesmo na actualização da amostra I_m, ou seja, para os cálculos a efectuar num determinado nível, só são necessários os resultados obtidos no nível imediatamente anterior. Desta forma, um qualquer raio não é mais necessário a partir do momento que a amostra respectiva tenha sido actualizada e que os raios seus descendentes tenham sido criados. Por conseguinte, ao ser destruído, não dá lugar a qualquer tipo de repetição de cálculos.

4.3.1 Organização por Blocos

A organização modular de uma implementação possível usando os pressupostos derivados da equação 4.19 apresenta-se em [Sou90] e reproduz-se na figura 4.7. Segue-se uma explicação breve das funções dos vários blocos que a constituem.

Cada raio luminoso é testado contra os objectos em cena a fim de se determinar qual o objecto intersectado mais próximo. Para o efeito, o bloco Determinação de Intersecções necessita de acesso à base de dados que contém a descrição dos objectos; recebe uma caracterização geométrica do raio (direcção e ponto de partida) e fornece uma caracterização completa da intersecção encontrada: objecto intersectado, ponto de intersecção e normal à superfície do objecto no ponto de intersecção.

O bloco Processamento de Intersecções identifica a superfície do objecto intersectado e, de acordo com a descrição da mesma na respectiva base de dados, actualiza a amostra correspondente no ecrã. Esta actualização contempla unicamente a iluminação ambiente determinada no ponto de intersecção (parcela $f_m.k_{a,m}.I_a$ da equação 4.19).

Com a descrição geométrica do raio intersectado e da intersecção e com as propriedades ópticas da superfície intersectada, o bloco Criação de Novos Raios gera os raios reflectido e transmitido e, através do acesso à base de dados de fontes de luz, os raios sensores de sombra. De acordo as equações 4.15 e 4.17, associam-se-lhes os respectivos factores de influência.

Todos os novos raios, incluindo a informação dos factores de influência, são depositados no Depósito de Raios de onde sairão quando forem seleccionados pelo bloco de Selecção de Raios, de acordo com a noção de realismo crescente e com os Parâmetros de Controlo respectivos, definidos interactivamente pelo utilizador.

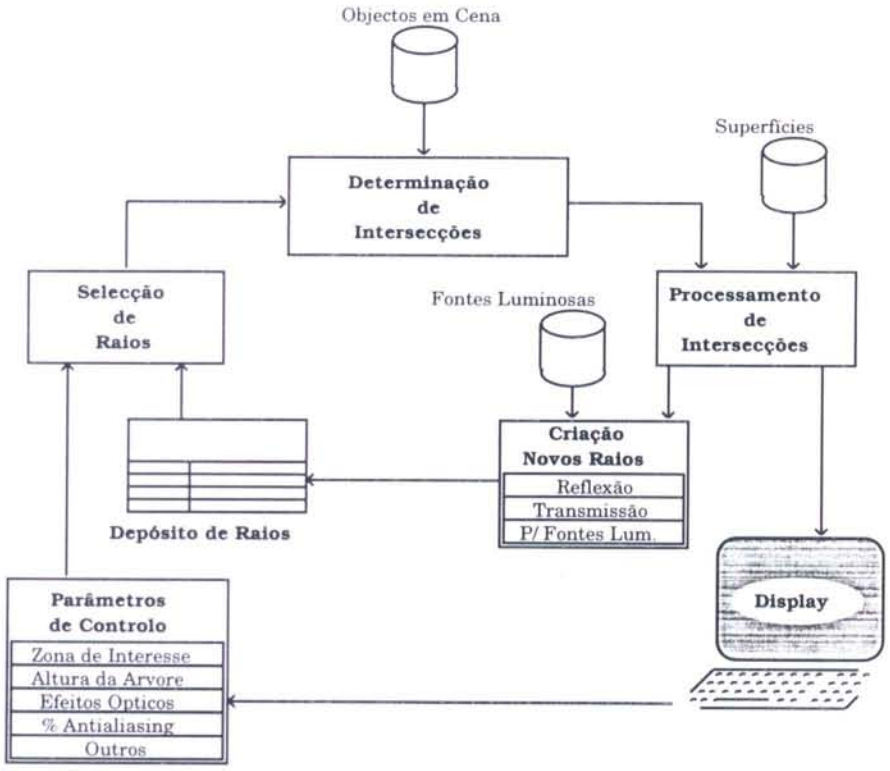


Fig. 4.7 - Organização modular de um sistema de *ray-tracing* com realismo crescente

Esta organização e funcionalidade dos principais blocos facilita uma maior interactividade do que na versão anterior [4.2], penalizada que era por um lado, pela repetição de cálculos efectuados (aumento do tempo de processamento) e, por outro, pela estratégia de funcionamento por passos. É agora possível a alteração, a qualquer momento e por parte do utilizador, dos parâmetros de controlo do realismo crescente, observando-se de imediato o seu efeito.

4.3.2 Controlo do Realismo Crescente

Os parâmetros de controlo do realismo crescente são sensivelmente os mesmos utilizados na versão descrita na secção [4.2] (dimensões de células imagem, profundidade da árvore de iluminação e definição de áreas de interesse) com algumas adições.

De acordo com a equação 4.19, a cada raio é associado um factor de influência que determina a percentagem da sua intensidade que atinge a amostra (ecrã) respectiva. Os raios com maior factor de influência tendem a colocar mais intensidade nas amostras respectivas e portanto a aumentar o seu brilho. Sob o ponto de vista de realismo

crescente, os raios com menor factor de influência têm, à partida, uma menor participação no brilho da imagem e portanto, o seu processamento pode ser retardado. É assim justificado o aparecimento de mais um parâmetro de controlo do realismo crescente directamente relacionado com o factor de influência.

Com respeito à profundidade da árvores de iluminação, algumas alterações decorrentes da experiência adquirida e permitidas pelo novo método são introduzidas.

Num contexto de realismo crescente, as primeiras aproximações de uma imagem servem fundamentalmente para o utilizador determinar a posição dos objectos. Não é de todo necessário, portanto, que contenham todos os fenómenos ópticos do modelo de iluminação e, nesta perspectiva, a transparência dos objectos, traduzida pelo modelo de iluminação como transmissão de raios luminosos, pode ser ignorada apesar de alguma perda de informação visual[1]. Por este motivo, a limitação da profundidade máxima da árvore de iluminação divide-se agora em duas componentes, uma para subárvores com ramos exclusivamente reflectidos e outra para subárvores contendo ramos transmitidos.

Tal como na versão anterior, os parâmetros de realismo crescente são interpretados como valores limites das grandezas a que estão associados: a profundidade na árvore é um valor máximo (dado que se trata de uma grandeza crescente ao longo do processo de crescimento do realismo), as dimensões das células são valores mínimos [4.2.3].

O parâmetro factor de influência relaciona-se com a grandeza com o mesmo nome que é associada com cada raio e que, segundo as equações 4.15 e 4.17 é uma grandeza positiva, tendencialmente decrescente com a profundidade na árvore (deriva da lei da conservação da energia que, os valores k, que representam coeficientes de reflexão e de transmissão, têm valores positivos inferiores à unidade). Este parâmetro corresponde, portanto, a um limite mínimo.

Em resumo, os parâmetros de controlo do realismo crescente são:

- factor de influência mínimo,
- profundidade máxima da árvore de iluminação para raios reflectidos e sem ascendência por transmissão,
- profundidade máxima da árvore de iluminação para raios transmitidos ou com ascendência por transmissão,
- dimensões mínimas das células imagem,
- definição de uma zona de interesse.

[1] Esta perda de informação visual pode ser comparada à que se obtém com a técnica de maquetes modeladas em barro, vulgarmente utilizada, por exemplo, pelos *designers* de automóveis.

Ao utilizador é assim dada a possibilidade de escolher se pretende avançar em profundidade e em factor de influência, mantendo células imagem de grandes dimensões ou se, pelo contrário, pretende manter uma profundidade baixa e diminuir as dimensões das células. A delimitação de áreas de interesse pode ser utilizada para modificar os parâmetros numa área mais restrita para a observação de pormenores.

4.3.3 Sessão de Utilização Típica

O avanço em profundidade na árvore de iluminação atribui à imagem mais efeitos ópticos, enquanto que a evolução nas dimensões das células aumenta a resolução, melhorando essencialmente o desenho de contornos.

Numa sessão típica de utilização das facilidades propostas, o utilizador prefere, para as primeiras imagens, localizar os objectos maiores, independentemente dos respectivos efeitos ópticos; neste ponto, árvores de nível 1, factores de influência elevados e uma resolução média são suficientes. Para a localização de objectos de menores dimensões, deve ser melhorada a resolução, pelo que o parâmetro a alterar deverá ser obviamente a dimensão mínima das células imagem.

Se temporariamente o utilizador se encontra satisfeito com a resolução que dispõe e, ainda que grosseiramente, pretende visualizar alguns reflexos e transparências, fixa as dimensões mínimas das células e aumenta o nível máximo das árvores de iluminação e, eventualmente o factor de influência mínimo. Uma vez obtida uma boa aproximação à imagem, a observação de detalhes localizados, sejam eles de resolução ou de efeitos ópticos, é possível pela definição de uma área de interesse, dentro da qual se introduzem melhoramentos na parametrização.

A principal desvantagem do algoritmo *ray-tracing* tradicional sob o ponto de vista de utilizador é o tempo que este perde, esperando por uma imagem refinada ao máximo, para, eventualmente, concluir que um objecto está mal colocado ou que uma sombra inesperada atira para segundo plano os objectos mais importantes e que, por consequência, a imagem não tem qualquer interesse.

Com as facilidades enunciadas para o sistema de *ray-tracing* com realismo crescente, essa situação pode ser evitada. Realmente, ao utilizador é dada a oportunidade de acompanhar e controlar a evolução da imagem, podendo, a qualquer momento, abortar o processamento.

A situação inversa também deve ser possível. Em qualquer momento, mesmo em estágios bastante precoces, o utilizador pode concluir que a imagem corresponde ao que pretende e deixa de acompanhar a sua evolução.

As duas afirmações anteriores conduzem a uma utilização do sistema em duas fases distintas. Na primeira, interactiva, o sistema sujeita-se às ordens dadas pelo utilizador através dos parâmetros de realismo crescente e selecciona para processamento os raios que mais influenciam a qualidade da imagem. Na segunda fase, não interactiva, a ordem de processamento de raios passa a ser completamente arbitrária, dado que o progresso da qualidade da imagem pode ser qualquer.

4.3.4 A Base de Dados de Raios

Os blocos de selecção e de depósito de raios são os mais sensíveis na organização descrita do sistema. É através deles que os raios vão sendo processados da forma mais adequada para elevar a qualidade da imagem, sempre dentro dos limites definidos pelos parâmetros de controlo definidos pelo utilizador.

O universo de raios disponíveis no bloco de armazenamento é grande em variedade. Inclui raios:

- de vários tipos (reflectidos ou transmitidos e sensores de sombra).
- com antecedentes diferentes (com ou sem transparências),
- de várias amostras (correspondendo a células imagem com dimensões diferentes),
- de vários níveis na árvore de iluminação,
- com valores de factores de influência variados.

Uma solução possível para o armazenamento e selecção de raios passa pela criação, no depósito respectivo, de uma base de dados ordenada por chaves, sendo estas relacionadas com os itens anteriores. Ao utilizador cabe então definir de que modo pretende a evolução de qualidade da imagem, actuando sobre as prioridades das várias chaves de ordenação.

A ordenação de tal base de dados deve ser efectuada de acordo com as prioridades estabelecidas pela noção de realismo crescente. Genericamente e atendendo às grandezas que controlam o realismo crescente, a prioridade de um raio é tanto maior quanto:

- maior for o factor de influência que lhe está associado,
- menor for o nível da árvore de iluminação em que se situa,

- maior for a célula imagem a que corresponde.

No entanto, tal estratégia só será possível se a quantidade de raios a ordenar for em número razoavelmente limitado. Realmente, o carácter eminentemente explosivo das árvores de iluminação faz antever um crescimento exponencial da quantidade de raios, o que cria problemas tanto na eficiência da algoritmia de ordenação da base de dados de raios como até mesmo, e talvez até mais grave, na quantidade de memória necessária para o seu armazenamento.

Parameterized Ray-Tracing [Seq89] é uma versão de *ray-tracing* cujo objectivo principal é utilizar a informação de intersecções raios/objectos resultante do cálculo de uma imagem para calcular rapidamente outras imagens da mesma cena com algumas alterações em (apenas) parâmetros relacionados com os efeitos ópticos dos objectos (coeficientes de reflexão, transmissão...). Partilha com o sistema de realismo crescente o problema da quantidade de memória que necessita para memorizar as árvores de iluminação.

Minimiza o problema por recurso a uma estrutura de dados particular, que contém, de uma forma compacta, as árvores de iluminação de todas as amostras de uma imagem. Algumas técnicas são utilizadas para a necessária compactação, baseadas quase todas em coerência (coerência da imagem, coerência horizontal de objectos, coerência vertical de objectos).

A estrutura que apresentam não é, no entanto, a mais conveniente para a versão de *ray-tracing* com realismo crescente. Por um lado, o seu objectivo principal é memorizar informação obtida de árvores processadas até ao seu limite, enquanto que, em realismo crescente, o problema é armazenar informação de raios ainda não processados. Por outro lado, e talvez mais importante, a forma algo aleatória com que os raios são processados em *ray-tracing* com realismo crescente, não garante formas de coerência que permitam o mesmo grau de compactabilidade da estrutura.

Questões semelhantes se podem colocar relativamente ao método *Incremental Ray-Tracing* [Mur90], destinado a acelerar os cálculos de uma sequência de animação por reutilização da informação de intersecções determinada em imagens anteriores.

Torna-se assim importante reduzir globalmente o universo de raios a processar e, especialmente, o conjunto de raios a manter ordenados, o que pode ser realizado com:

- uma utilização adequada dos parâmetros de controlo do realismo crescente, em conjunto com o faseamento do processo de realismo crescente em fase interactiva e fase não interactiva,

- a exploração de algumas propriedades, a enunciar, que os raios sensores de sombra apresentam.

4.3.4.1 Minimização da Base de Dados de Raios

Em [Sou90] e [Sou90a] mostra-se que, devido ao carácter recursivo do algoritmo *ray-tracing* tradicional, o número de raios aguardando processamento é proporcional à profundidade máxima da árvore de iluminação, sendo portanto bastante limitado.

Em *ray-tracing* com realismo crescente, a visita às árvores de iluminação efectua-se tendencialmente por níveis. Na situação de todos os raios processados gerarem um raio reflectido e outro transmitido, a quantidade de raios aguardando processamento duplica ao passar de um nível para o seguinte. Assim, uma árvore de iluminação completa com profundidade m em todas as $H.L$ amostras (ecrã), para uma cena incorporando N_S fontes de luz exige a memorização de N_R raios aguardando processamento tal que:

$$N_R = \begin{cases} H \cdot L & , m = 1 \\ H \cdot L \cdot [2^{(m-2)} \cdot (2 + N_S)] & , m \geq 2 \end{cases} \qquad \text{Eq. 4.20}$$

Note-se que, apesar dos resultados desta equação serem verdadeiramente incomportáveis, ela corresponde à pior situação possível que é todos os objectos em cena serem transparentes e, por conseguinte, gerarem, por cada intersecção, um raio reflectido e outro transmitido. No entanto, mesmo aceitando como mais próxima da situação real, uma árvore semicompleta do tipo da representada na figura 4.3 (onde não estão representados os raios sensores de sombra), a quantidade de raios aguardando processamento ainda seria:

$$\begin{aligned} N_R &= N \cdot L \cdot \left[m + (m-1) \cdot N_S \right] \\ &= N \cdot L \cdot \left[(m-1) \cdot (1 + N_S) + 1 \right], \qquad m \geq 1 \end{aligned} \qquad \text{Eq. 4.21}$$

Torna-se assim importante diminuir a quantidade total de raios a armazenar e, além disso, minimizar a quantidade de raios a manter ordenada na base de dados.

4.3.4.2 Partição da Base de Dados

Como se referiu na secção [4.3.2], os parâmetros de realismo crescente funcionam como valores limites das grandezas respectivas. Também se definiram duas fases distintas para uma utilização do sistema [4.3.3], a segunda das quais não exige qualquer ordem no processamento de raios. A interpretação conjunta destas duas questões permite diminuir a quantidade de raios a manter ordenados na base de dados.

De acordo com as características de cada raio e com os parâmetros de controlo do realismo crescente, o universo de raios $\mathcal{R}$ contido no depósito de raios durante a fase interactiva pode ser decomposto em dois subconjuntos:

$\mathcal{R}_g$ o conjunto de raios cujas características estão aquém dos valores definidos pelos parâmetros de realismo crescente,

$\mathcal{R}_{ng}$ o conjunto dos restantes raios.

O facto de, no conjunto $\mathcal{R}_{ng}$, as características de vários raios excederem os valores limites correspondentes aos parâmetros de controlo significa que, para o utilizador, esses raios não têm qualquer importância para a composição da imagem durante a fase interactiva. Sendo assim, podem ser colocados numa área de memória diferente da base de dados ordenada, aguardando que a fase não interactiva se inicie. Obviamente, a eficiência da ordenação da base de dados do conjunto $\mathcal{R}_g$ só tem realmente efeito se for acentuada a redução na quantidade de raios a ordenar.

Na tentativa de quantificar essa redução, suponha-se que o parâmetro dimensões de célula imagem possui o valor 4x4 *pixels*. Então, só uma amostra em cada dezasseis é processada, isto é, o conjunto $\mathcal{R}_g$ corresponde a somente 6.25% do conjunto $\mathcal{R}$.

No caso dos parâmetros de controlo da profundidade da árvore de iluminação, pode utilizar-se a menor prioridade dos raios transmitidos ou com ascendência por transmissão para reduzir ainda mais o conjunto $\mathcal{R}_g$ [4.3.2].

Para a demonstração da equação 4.20, partiu-se de uma árvore de iluminação completa até ao nível máximo $m=m_r=m_t$ (m_r e m_t são, respectivamente, a profundidade máxima para ramos contento exclusivamente reflexões e a profundidade máxima para ramos contendo transmissão ou com antecedentes por transmissão).

Suponha-se agora que os valores m_r e m_t são diferentes. A árvore de iluminação resulta desequilibrada e toma um aspecto semelhante ao que se representa na figura 4.8 (os raios representados a traço forte aguardam processamento).

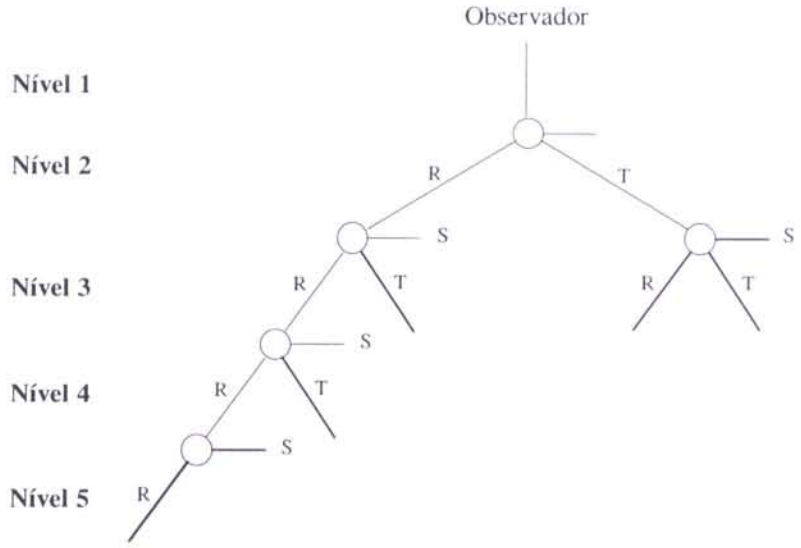


Fig. 4.8 - Diferentes níveis máximos, de reflexão e de transmissão, resultam numa árvore de iluminação desequilibrada

Desta forma, até ao nível m_t, a árvore de iluminação é completa. Para os níveis seguintes, os raios transmitidos ou com ascendência por transmissão ficam aguardando processamento, pelo que a árvore resulta bastante incompleta (no exemplo da figura 4.8, m_r=5, m_t=3). A equação que determina o número de raios memorizados aguardando processamento é agora:

$$N_R = H \cdot L \cdot [2^{(m_t-2)}.(2+N_S)+(m_r-m_t)] \qquad \text{Eq. 4.22}$$

Admitindo, como se viu anteriormente, que nas primeiras aproximações da imagem, as transmissões de raios têm pouca importância visual, então o nível m_t pode ser mantido com um valor bastante baixo enquanto que m_r pode tomar um valor superior. Neste caso e atendendo à equação 4.22, o número de raios aguardando processamento fica assim substancialmente reduzido relativamente ao da equação 4.20. Adicionalmente e dado que os raios com processamento suspenso não geram descendência, esta estratégia tem ainda a vantagem de diminuir a quantidade total de memória necessária para a base de dados de raios.

Em conclusão, a base de dados de raios que constitui os blocos depósito e selecção de raios pode ser subdividida em duas, uma delas contendo os raios considerados importantes para o crescimento do realismo durante a fase interactiva e outra contendo os restantes raios. Das duas, somente a primeira, de menores dimensões, necessita de ser ordenada.

Por semelhança com as duas fases com que se relacionam, as bases de dados designam-se respectivamente por Base de Dados Interactiva de raios (BDI) e Base de Dados Não Interactiva de raios (BDNI). Os raios que cada uma contém designam-se por Raios em Modo Interactivo (RMI) e Raios em Modo Não Interactivo (RMNI).

Durante uma sessão normal de utilização do sistema, cada novo raio criado no bloco de criação de raios é imediatamente classificado em RMI ou RMNI de acordo com os valores dos parâmetros de controlo do realismo crescente que estiverem definidos no momento.

4.3.4.3 Caso Especial: Raios Sensores de Sombra

Durante todo o processo de criação de uma imagem por *ray-tracing*, por cada raio processado, reflectido ou transmitido, geram-se dois novos raios por reflexão e por transmissão e N_S raios sensores de sombra. A quantidade destes últimos tende portanto a ser grande pelo que é de todo o interesse reduzi-la no sentido de diminuir a quantidade de memória necessária à implementação da base de dados de raios e a optimizar a eficiência dos procedimentos de manutenção da BDI ordenada.

Os raios sensores de sombra são de grande importância para a qualidade da imagem, como se pode comprovar da análise da equação 4.8, onde o seu efeito sobre a amostra é traduzido pela parcela em forma de somatório: na hipótese de esta parcela ser eliminada (ou o seu cálculo retardado) e de o processamento dos raios reflectido e transmitido ser retardado (ambiente de *ray-tracing* com realismo crescente), restaria a parcela correspondente à iluminação ambiente. Sendo esta parcela constante e independente de qualquer direcção, não oferece qualquer tipo de pista visual que confira a noção de volume ao objecto [Ama87a]. Em resumo, na ausência de raios sensores de sombra, a imagem de qualquer objecto é planar (por exemplo, a imagem de uma esfera resulta igual à de um disco) e portanto muito afastada da imagem realista, pelo que os raios sensores de sombra devem ter prioridade absoluta.

Sem grande prejuízo das afirmações feitas anteriormente acerca das prioridades estabelecidas pela noção de realismo crescente, podem, ou mesmo devem, colocar-se os raios sensores de sombra num estado de prioridade absoluta sobre os outros raios.

Por outro lado, os raios sensores de sombra gozam de uma propriedade importante que os diferencia dos raios reflectidos e transmitidos: não geram descendência. Face a esta propriedade, a quantidade de raios sensores de sombra só aumenta quando existe processamento de raios reflectidos ou transmitidos.

Assim, na hipótese de os raios sensores de sombra terem prioridade absoluta, nenhum outro raio tem oportunidade de ser seleccionado para processamento enquanto existirem sensores de sombra, e portanto não são criados mais raios deste tipo. A quantidade de raios sensores de sombra diminui até ao seu desaparecimento total da base de dados, momento em que um raio de outro tipo será seleccionado e processado, gerando uma descendência que incluirá N_S novos raios sensores de sombra, repetindo-se então o ciclo.

Em conclusão, a estratégia de atribuir prioridade máxima, na base de dados, aos raios sensores de sombra, limita a quantidade destes na base de dados a um número que iguala a quantidade N_S de fontes de luz em cena[1].

Note-se que esta é a quantidade máxima de sensores de sombra que se encontram simultaneamente na base de dados. O total de raios deste tipo a processar durante o cálculo de uma imagem completa é muito elevado (de crescimento exponencial) o que obriga a constantes operações de inserção e extracção da base de dados.

O facto de os raios sensores de sombra terem prioridade absoluta permite evitar os tempos de acesso à base de dados ordenada para inserção e extracção de raios sensores de sombra. Para o efeito, a BDI subdivide-se novamente em duas partes, uma BDIss que contém os raios sensores de sombra, e outra BDIrt que contém os raios RMI reflectidos e transmitidos.

A quantidade de raios simultaneamente na BDIrt é agora, partindo da equação 4.22:

$$N_R = H \cdot L \cdot \left[2^{(m_t-1)} + (m_r - m_t)\right] \qquad \text{Eq. 4.23}$$

Quanto à base de dados BDIss, a quantidade de raios que contém é normalmente muito pequena (N_S). O seu processamento efectua-se num espaço de tempo tão curto que o utilizador não consegue acompanhar a sua evolução raio a raio. Por este motivo, a BDIss não necessita de ser ordenada.

4.3.4.4 A Base de Dados de Raios Ordenada

A Base de Dados de Raios BDIrt contém os raios considerados importantes para a qualidade da imagem durante a fase interactiva de uma sessão de utilização do sistema. Com foi visto anteriormente, a importância destes raios, para a qualidade da imagem, não

[1] Como se verá mais tarde, a arquitectura paralela utilizada faz aumentar esta quantidade para um número que, no entanto, se mantém perfeitamente limitado.

é a mesma e, de acordo com a noção de realismo crescente, deve ser estabelecida uma ordem pela qual os raios são processados. Por este motivo, a BDIrt é mantida ordenada de acordo com os valores das grandezas relacionadas com os parâmetros de realismo crescente que caracterizam cada raio.

Em regime estável do sistema, isto é, para um determinado conjunto de parâmetros de controlo do realismo crescente, uma operação frequente a efectuar sobre a base de dados é a extracção de um raio que se considera ser, no momento, o mais conveniente para o progresso da qualidade da imagem. Supondo a base de dados ordenada por ordem decrescente de importância dos raios, esta operação corresponde à extracção do elemento com maior prioridade.

Esse elemento, uma vez extraído, é processado e gera descendência constituída por N_S raios sensores de sombra, um raio reflectido e, eventualmente, um raio transmitido. Os sensores de sombra são colocados directamente na BDIss e, como se viu, não são ordenados. Os raios reflectido e transmitido são inseridos na BDNI ou na BDIrt e, no interior desta, devem ficar localizados, relativamente aos restantes, de acordo com a sua prioridade.

As duas operações anteriores correspondem efectivamente à inserção e extracção de elementos numa estrutura de informação do tipo *larger in/first out* ou seja, numa fila de prioridade.

Em regime transitório, uma terceira operação é ainda necessária. De acordo com a descrição efectuada das facilidades do sistema, o utilizador pode, a qualquer momento, alterar os parâmetros de controlo do realismo crescente. Neste caso, é necessária uma reclassificação dos raios depositados nas bases de dados BDIrt e BDNI, isto é, raios que, face aos parâmetros anteriores, eram classificados como RMI podem passar a RMNI e vice-versa.

Obviamente, a reclassificação anterior obriga à reordenação de todos os elementos depositados na BDIrt.

Uma das formas mais eficientes de implementação de filas de prioridade é a árvore de prioridades que, além disso, é facilmente representável por um vector. Este último aspecto é simpático, dado que a linguagem de programação utilizada é OCCAM2 que não possui apontadores para memória.

Segundo a bibliografia especializada, nomeadamente [Knu73], a inserção ordenada e a extracção do elemento mais prioritário, na árvore de prioridade de raios que constitui a BDIrt, são operações de complexidade de ordem $O(\log N_{Irt})$ em que N_{Irt} representa o número de raios na BDIrt. Estas operações são portanto bastante optimizadas, não

colocando qualquer tipo de problemas à eficiência do sistema, até porque, na arquitectura utilizada, podem ser efectuadas em paralelo com os restantes cálculos.

Já a operação de ordenação a efectuar após a reclassificação de raios tende a ser menos eficiente. Segundo as mesmas fontes, a melhor complexidade que se consegue é de ordem $O(N_{Irt} . \log N_{Irt})$ o que pode causar alguns atrasos. Estes atrasos reflectem-se na forma de paragens aparentes do sistema sempre que os parâmetros de controlo do realismo crescente sejam alterados o que, num sistema que se pretende interactivo, não é desejável.

A única forma de melhorar a eficiência da classificação e ordenação dos raios na base de dados é portanto a minimização do valor N_{Irt}, o que justifica o esforço referido na secção [4.3.4.1] para a classificação dos raios em RMI e RMNI e respectiva colocação em bases de dados distintas, BDIrt e BDNI.

4.3.5 Paralelização do Sistema

Um dos pressupostos do trabalho a que este texto diz respeito, é o estudo e desenvolvimento de uma arquitectura paralela para síntese de imagens por meio de *ray-tracing* com realismo crescente. As arquitecturas paralelas para *ray-tracing* são uma via de investigação importante e com um potencial muito grande face aos últimos avanços verificados na área das arquitecturas paralelas.

Surgidas na década de oitenta, as arquitecturas baseadas em *transputers* atingem, sensivelmente na época em que este trabalho se inicia, uma grande popularidade na comunidade científica. A sua modularidade, facilidade de expansão e desempenho uniprocessador, aliados a um custo comercial aceitável e denotando tendência para uma forte descida, são factores decisivos importantes que justificam a sua selecção no contexto deste trabalho.

A ideia original presente nas especificações do trabalho define-o como um subsistema autónomo de síntese de imagem que possa ser interligado com um sistema de desenho assistido por computador, complementando as facilidades de visualização do mesmo e fornecendo ao utilizador a hipótese de observar a evolução das imagens em realismo crescente. Três processos distintos são assim necessários:

- Interacção,
- Cálculo,
- Visualização.

O processo de Interacção, na perspectiva de realismo crescente, permite ao utilizador controlar a forma como a qualidade da imagem evolui com o tempo e estabelece os acessos às unidades de informação necessárias para a leitura da descrição das cenas a visualizar e para a colocação das imagens calculadas.

O processo de Cálculo, o mais necessitado de capacidade de processamento, corresponde a uma rede de processadores em paralelo. Efectua cálculos de *ray-tracing* em modo realismo crescente, atendendo às especificações recebidas do módulo de interacção e envia os resultados para o processo de visualização.

O processo de Visualização actualiza as imagens e afixa-as no ecrã para visualização imediata. Por razões que se prendem com a capacidade de processamento e de transmissão de informação, este módulo é realizado em arquitectura do mesmo tipo do processo de cálculo.

4.3.6 Os Processos do Sistema

Os três processos mencionados possuem correspondência directa sobre os blocos representados na figura 4.7. O processo de Interacção corresponde ao bloco Parâmetros de Controlo e o de Visualização ao bloco com o mesmo nome. O processo de Cálculo relaciona-se com os restantes blocos pelo que pode ainda subdividir-se em subprocessos de acordo com as suas funcionalidades: Manutenção da Base de Dados (blocos Depósito e Selecção de Raios) e Cálculo de Intersecções.

Um processo adicional, Coordenação, garante um fluxo de informação adequado entre os outros processos e integra, da figura 4.7, o bloco Geração de Novos Raios. Realmente, a informação de entrada neste bloco é a informação de intersecção que o processo Coordenador recebe do processo Cálculo de Intersecções. Dado que as operações a efectuar pelo bloco Geração de Novos Raios são simples e rápidas, justifica-se a integração.

A figura 4.9 representa os processos mencionados e as respectivas ligações para efeitos de comunicação [Sou95].

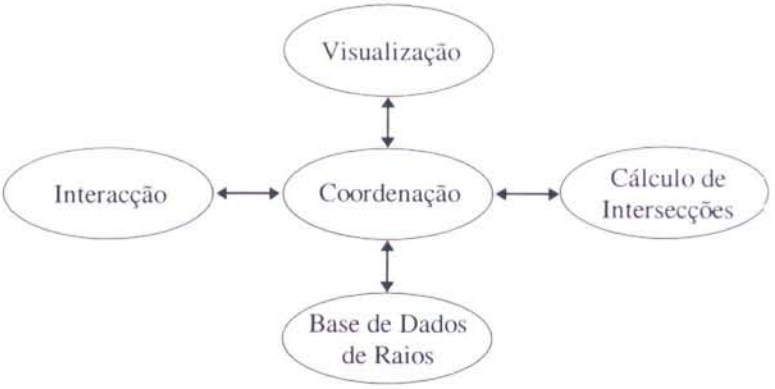


Fig. 4.9 - Os processos principais do sistema

O processo Interacção implementa basicamente uma interface com o utilizador do tipo menu. O processo Coordenação, tal como o nome indica, coordena os restantes processos e dá seguimento aos comandos dados pelo utilizador. O processo Base de Dados de Raios memoriza os raios aguardando processamento de acordo com as estratégias vistas anteriormente. O processo Cálculo de Intersecções corresponde realmente a vários processos paralelos, em processadores diferentes, de forma a acelerar os respectivos cálculos. O processo Visualização recebe informação de raios e intersecções vinda do coordenador e actualiza a imagem de acordo.

4.3.6.1 Estratégia de Paralelização

Como se afirmou na secção [1.2.1], a maior parte do tempo consumido em *ray-tracing* corresponde ao cálculo de intersecções de raios com objectos. Face à estrutura de processos descrita para o sistema, é portanto de esperar que o maior esforço de paralelização seja efectuado no processo Cálculo de Intersecções que é implementado sobre um conjunto de vários processadores.

Em secção anterior [2.6.1], apresentou-se a taxonomia de S. Green [Gre91] relativa à paralelização de algoritmos de *ray-tracing*. De acordo com a implementação em realismo crescente, as estratégias de paralelização Subdivisão Imagem a Imagem e Vectorização não são adequadas. A Subdivisão no Espaço Objecto é possível mas exige esquemas complicados de distribuição do espaço ou de objectos de forma a obter um balanceamento equilibrado da carga computacional. A Subdivisão no Espaço Imagem apresenta-se com a alternativa mais óbvia, embora careça de algumas considerações adicionais.

Realmente, na presente implementação de *ray-tracing* com realismo crescente, a paralelização não se efectua ao nível de áreas do ecrã mas sim ao nível dos raios, quaisquer que eles sejam. No entanto, a seguinte interpretação pode ainda ser dada: um raio é, a dado momento, a entidade em processamento que caracteriza uma determinada célula imagem. Assim, ao paralelizar o processamento de raios, paraleliza-se implicitamente o processamento de diferentes áreas do ecrã.

A subdivisão no espaço imagem possibilita um balanceamento equilibrado da carga computacional, principalmente se é gerido dinamicamente. Na actual implementação, a granularidade das tarefas a executar é muito fina o que reforça ainda mais esta noção. No entanto, o quociente Comunicações/Processamento é mais elevado pelo que a eficiência da paralelização tende a baixar quando o número de processadores aumenta [Sou95].

Aliás, o peso das comunicações influencia também a escolha da forma de acesso à base de dados de objectos. Segundo S. Green, aquela base de dados pode ser duplicada ou distribuída pelos processadores que dela necessitam. No entanto, a versão de distribuição aumenta ainda mais a quantidade de mensagens entre processadores, pelo que a duplicação parece ser, nesta fase experimental, a mais adequada. Desta forma, os processadores contidos no processo de Cálculo de Intersecções executam o mesmo código com os mesmos dados iniciais.

Note-se que, no entanto, a duplicação da base de dados de objectos limita o tamanho da mesma pela menor quantidade de memória encontrada em um dos processadores. Esta é uma restrição à complexidade das cenas que, numa fase experimental não parece demasiado grave. As bases de dados de superfícies e de fontes de luz, pelas suas muito menores dimensões, não criam problemas deste tipo.

Sob o ponto de vista da classificação dos mecanismos de decomposição de problemas [2.2] efectuada por A. Chalmers [Cha91], duas classificações podem ser efectuadas, consoante o nível de detalhe com que se observa o sistema. Ao nível mais elevado, o sistema, globalmente, corresponde a uma decomposição algorítmica *data flow*, dado que as várias componentes do algoritmo são executadas por processos diferentes. A um nível mais baixo, o processo Cálculo de Intersecções apresenta uma decomposição no domínio, orientada às tarefas: uma tarefa corresponde ao cálculo da primeira intersecção de um raio; os dados específicos da tarefa são as características do próprio raio; os outros dados são a caracterização geométrica dos objectos que compõem a cena.

Dado que a maior concentração de poder de cálculo em paralelo se situa no processo Cálculo de Intersecções, a componente *data flow* é desprezável e o sistema é visto como uma decomposição no domínio, orientada às tarefas. Uma das mais populares formas de paralelizar um problema com este tipo de decomposição é a *Farm* de processadores

[May89], segundo a qual um processo *Farmer* ou *Master* coordena a execução de várias tarefas semelhantes que são distribuídas por vários processos *Workers* à medida que estes ficam disponíveis. Sempre que um *worker* esteja disponível, um sinal é enviado ao *master* que lhe envia uma nova tarefa para executar. As comunicações necessárias acontecem somente entre o *master* e qualquer um dos *workers*.

No entanto, dadas as limitações em termos de quantidade de ligações físicas entre os *workers* e o *master*, é necessário partilhar essas ligações. Assim, estabelece-se uma rede de ligações entre *workers* que, por sua vez, comunica com o *master* por um número limitado de ligações. O problema desta solução é que, uma mensagem entre o *master* e um *worker* tem que atravessar vários processadores até atingir o seu destino. A transmissão de mensagens através de processadores intermédios consome tempo de CPU, pelo que técnicas de programação adequadas[1] devem ser seguidas por forma a minimizar o problema [Atk89].

Várias topologias são permitidas para a rede mencionada, com efeitos diferentes sobre o quociente Tempo de Comunicações/Tempo de Processamento [Pac89]. Em [Pur90] apresenta-se um estudo efectuado, sobre uma implementação em *Farm* de um algoritmo *ray-tracing* sobre uma arquitectura paralela baseada em *transputers*, para avaliar o comportamento de topologias em linha, árvore ternária e em anel. A conclusão do estudo é claramente a favor da árvore ternária, o que se justifica por apresentar a menor distância média entre o *master* e os *workers*.

Em conclusão, o sistema a que este texto se refere corresponde a uma implementação paralela, sobre uma arquitectura baseada em *transputers*, de um algoritmo *ray-tracing* com subdivisão no espaço imagem, duplicação da base de dados e balanceamento dinâmico de carga computacional. Apresenta uma decomposição no domínio, orientada às tarefas, segundo a estratégia em *farm* de processadores. Os *workers* distribuem-se segundo uma árvore ternária.

Na figura 4.10 apresenta-se a arquitectura utilizada, com indicação de processos (representados por círculos) e de processadores (representados por rectângulos). Note-se a situação dos processos Visualização e Base de Dados de Raios em processadores dedicados. O primeiro garante assim o acesso directo à memória de vídeo (a placa gráfica é também baseada em *transputers*) e o segundo permite o acesso a uma quantidade de memória superior à que é disponibilizada pelos restantes processadores do sistema.

[1] Técnicas de programação que, por meio de buferização, façam uma utilização exaustiva dos dispositivos de DMA afectos aos *links* do *transputer*.

Aliás, o recurso a um processador dedicado para a Base de Dados de Raios permite que a actualização desta, após as operações de inserção e de extracção de raios, se faça em paralelo com outras operações.

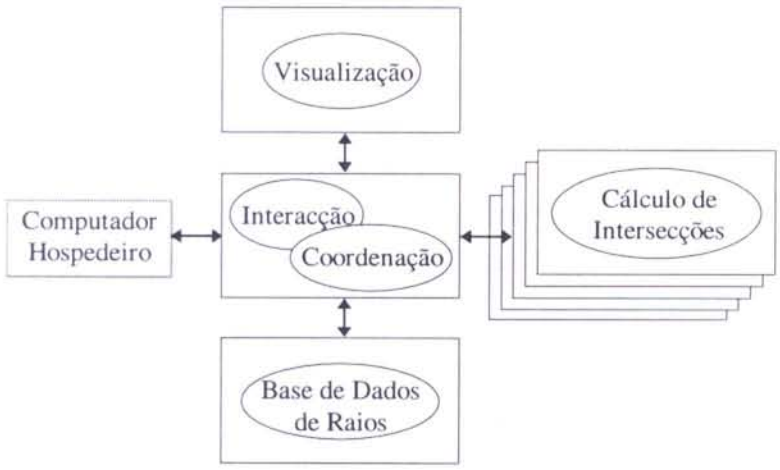


Fig. 4.10 - Arquitectura do sistema

4.3.6.2 Os *Workers* da *Farm*

Numa *farm* de processadores, de acordo com D. May [May89], cada *worker* processa uma tarefa, devolve o resultado ao *master* e recebe deste uma nova tarefa para executar. De acordo com esta afirmação, existem, para cada *worker*, dois percursos de informação: de e para o *master*.

Por outro lado, uma mensagem atravessa vários *workers* antes de atingir o seu destino. De forma a que os processadores intermédios não prejudiquem o fluxo de informação, cada *worker* é dotado de dois processos, em concorrência com a aplicação que lhe é devida, cuja função é gerir as comunicações com um mínimo de participação do CPU. Estes processos são conhecidos por *Routers:*

- *Router* de dados, responsável pelo percurso correspondente às tarefas,
- *Router* de resultados, responsável pelo percurso correspondente aos resultados.

Na solução preconizada por David May, a topologia da *farm* é do tipo em linha. Assim, cada *router* comunica com os dois *routers* do mesmo tipo residentes nos dois processadores que são seus vizinhos imediatos (anterior e posterior) e com a aplicação que corre no próprio processador. No caso da topologia em árvore, o número de ligações

posteriores aumenta. A figura 4.11 mostra os processos que correm num *worker* e as respectivas ligações.

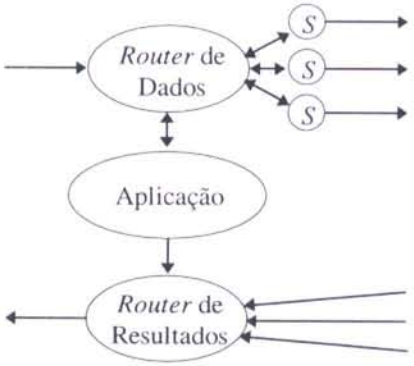


Fig. 4.11 - Os processos constituintes de um *worker*

Os processos *S* (*sender*) representados na figura garantem o regular funcionamento do *router* de dados, mesmo que uma das suas saídas fique bloqueada. Em ambos os *routers* foram utilizadas técnicas de *fair ALT*, baseadas na declaração de canais favoritos [Jon89] de forma a evitar situações de *starvation*.

Relativamente à descrição de *farm* apresentada por David May, a implementação efectuada tem a vantagem de reduzir os tempos de ócio (*idle*) nos *workers* nos momentos seguintes ao término de uma tarefa. Para o efeito, foi desenvolvida uma variante do método que, aproveitando a "bufferização" inerente aos *routers* de dados, pressiona as tarefas no sentido das aplicações locais e dos *routers* seguintes. O método é ainda corrector do balanceamento de carga computacional.

```
SEQ
  favourite, active := 0, TRUE
  WHILE active
    PRI ALT i=favourite FOR n.ways.in
      VAL k IS i REM n.ways.in:
      inputs[k] ? message
        SEQ
          output ! message
          favourite := (k + 1) REM n.ways.in
          ... active := FALSE ?
```

Fig. 4.12 - Pseudocódigo OCCAM do *router* de resultados

O pseudocódigo OCCAM do processo *router* de resultados é basicamente um *multiplexer* de canais e apresenta-se na figura 4.12. A constante *n.ways.in* representa o número de canais de entrada (um da aplicação local e os restantes dos processadores seguintes).

Todas as mensagens recebidas pelos canais de entrada são reenviadas para o processador anterior pelo canal *output*. A variável *favourite* em conjugação com a construção PRI ALT garante um atendimento justo de todos os canais de entrada (isento de *starvation*).

Os pseudocódigo OCCAM dos processos *sender* e principal do *router* de dados encontram-se nas figuras 4.13 e 4.14.

```
SEQ
  active := TRUE
  WHILE active
    SEQ
      sender.to.main ! TRUE
      inp ? message
      out ! message
      ...  active := FALSE ?
```

Fig. 4.13 - Pseudocódigo OCCAM do processo *sender*

Cada *router* de dados possui um conjunto de réplicas de *senders*. Sempre que um *sender* se encontre disponível, envia um sinal pelo canal *sender.to.main* que o processo principal recebe por um canal do vector *from.sender*. Identificado *sender.ready*, o processo principal do *router* envia para o mesmo processo *sender* uma mensagem recém-recebida do canal *from.left* que o liga ao processador anterior.

```
SEQ
  active, favourite:= TRUE, 0
  WHILE active
    from.left ? message
      PRI ALT i=favourite FOR n.ways.out
        VAL sender.ready IS i REM n.ways.out:
        from.sender[sender.ready] ? dummy
          SEQ
            into.sender[sender.ready] ! message
            favourite := (favourite + 1) REM n.ways.out
```

Fig. 4.14 - Pseudocódigo OCCAM do processo principal do *router* de dados

Em concorrência com os dois *routers*, a aplicação local concentra-se na determinação da primeira intersecção de cada raio que recebe (tarefa) do *router* de dados. Contém, para o efeito, uma cópia da base de dados de objectos. O resultado de cada intersecção é enviado para o *router* de resultados a partir do qual se encaminha para o processador *master* (processo Coordenação).

4.3.6.3 Coordenação

O processo Coordenação recebe os comandos dados pelo utilizador através do processo interacção e executa-os, desencadeando, quando necessário, um conjunto de comandos que distribui pelos outros processos. Tem a responsabilidade de coordenar os trabalhos desenvolvidos pelos outros processos à sua volta, sendo especialmente crítica a tarefa de manter um balanceamento de carga computacional equilibrado nos *workers* da *farm*, alimentando, para o efeito, o processo Cálculo de Intersecções com raios extraídos da Base de Dados. Recebe raios e respectivas intersecções do processo Cálculo de Intersecções, envia-os para o processo Visualização e cria novos raios[1] que entrega ao processo Base de Dados de Raios.

O processo Coordenação corresponde ao processador *farmer*, na definição de D. May. Tal como os *routers* de dados, comunica com o processador seguinte (primeiro *worker*) através de um processo *sender*. As suas funções são, como se viu, a coordenação dos restantes processos, nomeadamente no que respeita à gestão das tarefas a enviar para os *workers* e a execução dos comandos dados pelo utilizador. A figura 4.15 representa o pseudocódigo OCCAM do processo Coordenação, na fase interactiva do algoritmo.

Note-se, na figura, a ordem com que as guardas são declaradas na construção PRI ALT. A maior prioridade é dada ao canal procedente no processo Interacção de forma a garantir que, mesmo em regime de grande fluxo de informação, de e para os *workers*, o utilizador consegue interromper o sistema. Seguidamente, a maior prioridade é dada ao canal pelo qual se detecta a disponibilidade do *sender*, de forma a criar um mínimo de atrasos no envio de informação para os *workers*.

Durante a fase interactiva, o processo Coordenação alimenta o processo *sender* com raios que este enviará para os *workers*. Esta acção está no entanto condicionada, por um lado, pela disponibilidade do próprio *sender* e, por outro lado, pela existência de raios RMI na base de dados (DBIss ou BDIrt). Com as intersecções recebidas dos *workers*, enviam-se mensagens para o processo Visualização e criam-se os novos raios que são enviados para o processo Base de Dados de Raios.

Uma variável *rays.in.farm* mantém o processo informado acerca da quantidade de raios que se encontram na *farm* para cálculo de intersecções. Este valor é necessário para a decisão de modificar os parâmetros de controlo do realismo crescente, ou mesmo de terminar a fase interactiva do algoritmo (condição de *exit state*).

---

[1] A criação dos novos raios, reflectido, transmitido e sensores de sombra, é complementada pela afectação dos mesmos pelos factores de influência determinados pelas expressões 4.15 e 4.17.

```
IF
  sender.is.ready
    SKIP
  TRUE
    SEQ
      from.sender ? dummy
      sender.is.ready := TRUE
WHILE mode = inter.mode
  SEQ
    PRI ALT
      from.interaction ? user.message
        ...  execute user.message
      from.sender ? dummy
        sender.is.ready := TRUE
      ((NOT RaysDB.is.empty) AND sender.is.ready) & SKIP
        SEQ
          ...  ask RaysDB for one ray IMR;
          ...  RaysDB.is.empty := FALSE ?
          into.sender ! ray.message
          rays.in.farm := rays.in.farm + 1
      from.sb ? intersection.message
        SEQ
          rays.in.farm := rays.in.farm - 1
          into.visualization ! intersection.message
          ...  make.new.rays.to.RaysDB
    IF
      RaysDB.is.empty AND (rays.in.sb = 0)
        SEQ
          ...  if possible, update parameters
          ...  else, change to null mode
      TRUE
        SKIP
```

Fig. 4.15 - Pseudo-código OCCAM do processo Coordenação

Em primeira análise, essa decisão poderia ser tomada quando as bases de dados BDIss e BDIrt fossem encontradas vazias, não havendo portanto mais tarefas a executar na fase interactiva. No entanto, uma rede de *workers* composta por P processadores pode conter até $k.P$ raios em que k designa o número total de *buffers* associados aos *routers* e à aplicação de cada processador. Admitindo que nenhum dos $k.P$ raios é sensor de sombra, então a sua descendência elevar-se-ia a:

$$N_R = k \cdot P \cdot (2 + N_S) \qquad \text{Eq. 4.24}$$

Admitindo ainda a hipótese de todos os novos raios serem classificados como RMI (o que acontece pelo menos com os raios sensores de sombra), as bases de dados BDIss e BDIrt crescem imediatamente para, no mínimo, N_R raios, devendo o algoritmo manter-se, por consequência, na fase interactiva.

Em resumo, a condição expressa na última construção IF do pseudocódigo da figura 4.15 garante uma terminação correcta da fase interactiva do algoritmo, ou simplesmente uma alteração em tempo certo dos parâmetros de controlo do realismo crescente, com base na ausência total de raios, quer nas bases de dados interactivas de raios, quer na rede de processadores que actuam como *workers*.

4.3.6.4 Base de Dados de Raios

O processo Base de Dados de Raios inclui as três bases de dados mencionadas, BDIss, BDIrt e BDNI. Compete-lhe classificar e armazenar os raios que o processo Coordenação lhe entrega, assim como devolver-lhe, a pedido, um raio RMI ou RMNI.

Assim, classifica todos os raios que recebe em RMI (sensores de sombra ou não), ou RMNI, colocando-os na base de dados respectiva. A inserção na BDIrt é feita atendendo às prioridades dos raios para o realismo crescente.

Havendo lugar a alteração dos parâmetros de controlo do realismo crescente, o processo Coordenação comunica ao processo Base de Dados os novos valores que, por sua vez, reclassifica os raios das BDIrt e BDNI e reordena a refeita BDIrt. Esta operação é um pouco lenta, pelo que podem surgir alguns momentos de paragem aquando da reordenação.

O processo Base de Dados de Raios aceita dois tipos de pedidos de raios do processo Coordenação, de acordo com a classificação de raios efectuada: RMI ou qualquer raio. Os dois tipos de pedido são utilizados respectivamente durante a fase interactiva e durante a fase não interactiva do algoritmo.

Ao pedido de um raio RMI, a resposta é dada de acordo com as prioridades dos raios para o crescimento do realismo. Assim, por ordem decrescente de prioridade, o processo Base de Dados de Raios responde:

- raio sensor de sombra se BDIss não é vazia,
- raio reflectido ou transmitido se BDIrt não é vazia,
- sinal de erro (ausência de raios RMI).

Cabe ao processo Coordenação definir a acção seguinte quando lhe é comunicado o sinal de erro referido.

Os pedidos de raios RMNI são normalmente efectuados durante a fase não interactiva pelo que a ordem de prioridades de resposta pode ser qualquer. No entanto, por uma

questão de coerência com o modo interactivo, mantém-se a mesma ordem afectada somente pelo aparecimento de mais um tipo de resposta:

- raio sensor de sombra se BDIss não é vazia,
- raio reflectido ou transmitido se BDIrt não é vazia,
- raio reflectido ou transmitido se BDNI não é vazia,
- sinal de erro (ausência de raios RMI).

O pseudocódigo correspondente pode ser visto na figura 4.16.

```
active, ask.type := TRUE, NUL
WHILE active
  SEQ
    from.coord ? CASE
      add.ray; ray.mess
        ...  Classify Ray;
        .....insert in DBIss,DBIrt or DBNI accord.
      ask.interact.mode.ray
        ask.type := RMI
      ask.any.ray
        ask.type := any.RM
      param; param.rec
        SEQ
          ...  Reclassify rays DBIrt & DBNI;
          ...  exchange rays as necessary
          ...  Heap Sort DBIrt
    IF
      ask.type = NUL
        SKIP
      rays.in.DBIss > 0
        SEQ
          ...  Extract from DBIss to ray.mess
          rays.in.DBIss := rays.in.DBIss - 1
          into.coord ! ray.reply; ray.mess
      rays.in.DBIrt > 0
        SEQ
          ...  Extract from DBIrt to ray.mess
          rays.in.DBIrt := rays.in.DBIrt - 1
          into.coord ! ray.reply; ray.mess
      (ask.type = any.RM) AND (rays.in.DBNI > 0)
        SEQ
          ...  Extract from DBNI to ray.mess
          rays.in.DBNI := rays.in.DBNI - 1
          into.coord ! ray.reply; ray.mess
      TRUE
        into.coord ! error; DB.empty
```

Fig. 4.16 - Pseudocódigo do Processo Base de Dados de Raios

Na secção [4.3.4.3], mostrou-se que, atribuindo prioridade máxima aos raios sensores de sombra, a quantidade destes fica limitada a um número que iguala a quantidade N_S de fontes de luz em cena, o que é fundamental para o dimensionamento da BDIss. O número

encontrado deriva do facto de o bloco Determinação de Intersecções da figura 4.7 processar um raio de cada vez.

Na presente implementação, o bloco Determinação de Intersecções corresponde à rede de processadores que constituem os *workers* da *farm* que, de acordo com as descrições anteriores, determinam as intersecções de vários raios simultaneamente. Então, nas mesmas condições de Eq. 4.24, a quantidade de raios sensores de sombra pode crescer até um valor:

$$N_{SS} = k \cdot P \cdot N_S \qquad \text{Eq. 4.25}$$

Ou seja, na presente implementação paralela, a quantidade máxima de raios na BDIss é proporcional, não só ao número de fontes luz da cena, mas também ao número de processadores que efectuam a determinação de intersecções. O coeficiente de proporcionalidade é o número de *buffers* colocados em cada um destes processadores.

Em termos globais e como se mostrou na secção [4.3.4], o processo Base de Dados de Raios requer uma grande quantidade de memória. Por este motivo, este processo é colocado num processador dedicado que, no caso da arquitectura utilizada, possui 24M*Bytes* de memória.

4.3.6.5 Interacção

O processo Interacção é basicamente uma interface com o utilizador do tipo menu. Aceita comandos de inicialização, de acesso a disco para leitura de ficheiros com descrição de cenas ou com imagens já calculadas, e de acesso a disco para escrita de ficheiros com a última imagem calculada. Comandos mais direccionados para o *ray-tracing* com realismo crescente são a visualização e possível alteração dos parâmetros de controlo do realismo crescente e a comutação entre a fase interactiva e a fase não interactiva do sistema. É implementado em concorrência com o processo Coordenação no processador mais próximo do computador hospedeiro.

4.3.6.6 Visualização

O processo Visualização é responsável pela actualização da imagem à cadência do cálculo de intersecções. Para o efeito, é colocado num processador dedicado, com a mesma tecnologia do restante sistema (*transputer*) que tem acesso directo à memória

vídeo. Recebe, do processo Coordenação, o resultado do cálculo de intersecção de cada raio e actualiza a amostra respectiva, de acordo com a equação 4.19.

Considere-se, naquela equação, o efeito de, exclusivamente, um raio de nível m na árvore de iluminação. A equação fica:

$$I_m = I_{m-1} + f_m \cdot k_{a,m} \cdot I_a + \sum_{S=1}^{N_S} \left(f_{S,m} \cdot I_S \right) \qquad \text{Eq. 4.26}$$

A parcela em forma de somatório corresponde à participação de raios sensores de sombra ($f_{S,m}$ é o factor de influência característico do raio e I_S é o brilho da fonte de luz para a qual o raio se dirige) e a segunda parcela corresponde à participação de um raio reflectido ou transmitido (f_m é o factor de influência característico do raio, $k_{a,m}$ é o coeficiente de reflexão ambiente da superfície intersectada pelo raio e I_a é a intensidade de iluminação ambiente considerada para toda a cena).

Um raio recebido pelo processo Visualização pode ter intersectado ou não um objecto da cena, facto que é relatado pela descrição de intersecção que recebe, juntamente com o raio, do processo Coordenação. A existência de intersecção é tratada de forma diferente, para os raios sensores de sombra e para os raios reflectidos e transmitidos, facto que não se encontra contabilizado na equação 4.26.

No caso de um raio sensor de sombra, a actualização $f_{S,m}.I_S$ corresponde a iluminar o ponto origem do raio pela fonte de luz S. Ora, se uma intersecção é determinada para o raio em questão, então existe pelo menos um objecto que se interpõe entre aquele ponto e a fonte de luz e portanto a visibilidade δ entre as duas entidades é nula: o ponto encontra-se na sombra no que respeita à fonte S. Assim, a comparticipação $I_{S,m}$ de um raio sensor de sombra para a amostra a que corresponde é:

$$I_{S,m} = \begin{cases} f_{S,m} \cdot I_S & , \delta = 1 \\ 0 & , \delta = 0 \end{cases} \qquad \text{Eq. 4.27}$$

No caso de um raio reflectido ou transmitido, a parcela $f_m.k_{a,m}.I_a$ caracteriza, de acordo com a interpretação dada às equações 4.15 e 4.16, a iluminação ambiente do objecto que o raio intersecta. Quando nenhum objecto é intersectado, é usual, em *ray-tracing*, afectar o raio de um brilho constante atribuído ao fundo (*background*). Assim, a comparticipação $I_{RT,m}$ de um raio reflectido ou transmitido para a amostra a que corresponde é:

$$I_{RT,m} = \begin{cases} f_m \cdot k_{a,m} \cdot I_a & ,\delta = 1 \\ f_m \cdot I_{bk} & ,\delta = 0 \end{cases} \qquad \text{Eq. 4.28}$$

A determinação dos valores $k_{a,m}$ e I_a requer o acesso às bases de dados respectivamente de superfícies e de fontes de luz. (figura 4.7, bloco de Processamento de Intersecções). Estas bases de dados são residentes no processador que corre o processo Visualização e são-lhe transmitidas pelo processo Coordenação aquando da leitura do ficheiro que descreve a cena.

```
active := TRUE
WHILE active
  inp ? CASE
    upd.display.inters; ray&intersection.message
      IF
        Type.of.Ray(ray) = shadow.feeler
          SKIP
        TRUE
          VAL Fm          IS [field in ray]:
          VAL surface.n   IS [field in ray]:
          VAL surface.rec IS surfaces[surface.n]:
          VAL Kam         IS [field in surface.rec]:
          ...  update sample with Fm.Kam.Ia
    upd.display.no.inters; ray.message
      IF
        Type.of.Ray(ray) = shadow.feeler
          VAL Fsm       IS [field in ray]:
          VAL light.n   IS [field in ray]:
          VAL light.rec IS light.sources[light.n]:
          VAL Is        IS [field in light.rec]:
          ...  update sample with Fsm.Is
        TRUE
          VAL Fm IS [field in ray]:
          ...  update sample with Fm.Ibk
```

Fig. 4.17 - Pseudocódigo OCCAM do processo Visualização

4.3.7 Avaliação do Crescimento do Realismo

A avaliação do crescimento do realismo é efectuada com base na obtenção de imagens chave e respectivos tempos de processamento. De modo grosseiro, pode dizer-se que o sistema tem um comportamento aceitável se o tempo conseguido para uma imagem parcial, minimamente interpretável, for bastante inferior ao tempo total de processamento da imagem final.

Uma imagem chave corresponde ao estágio da imagem no limite de um determinado conjunto de parâmetros de controlo do realismo crescente. Por exemplo, se o parâmetro de controlo de dimensões de células imagem e o parâmetro de controlo de profundidade na árvore de iluminação se encontram definidos respectivamente em 4x4 e 2, então a qualidade da imagem aumenta à medida que as dimensões das células imagem diminuem (a partir de um valor especificado) e que a profundidade da árvore aumenta (a partir de zero) até uma imagem chave, que se obtém no momento em que as dimensões de todas as células em processamento atinjam 4x4 e a profundidade da árvore atinja o valor 2 nas amostras respectivas.

Conforme tem vindo a ser afirmado, cada célula imagem caracteriza-se por uma amostra. É importante nesta fase, determinar, face à arquitectura e algoritmia utilizadas, de que forma se pode efectuar essa caracterização, isto é, face à cor determinada para uma amostra segundo a equação 4.19, qual será a afectação da célula respectiva.

As células imagem podem ser [Sou92]:

- Células imagem uniponto
- Células imagem rectangulares
  - com iluminação constante
  - com iluminação suavizada

Uma célula imagem uniponto corresponde a uma determinada área do ecrã na qual somente um ponto no seu interior se encontra iluminado. Uma célula imagem rectangular corresponde a uma área idêntica, mas iluminada na sua totalidade. A iluminação da célula pode ser constante se a amostra é replicada em todos os seus pontos ou pode ser suavizada se, de acordo com essa amostra e com outras encontradas à sua volta, se efectua uma interpolação bilinear da iluminação.

Por outro lado, dado que o sistema é interactivo, a ordem pela qual as células são actualizadas no ecrã influencia, a cada momento, a avaliação que o utilizador faz da imagem. Embora as células imagem sejam actualizadas no ecrã seguindo (fundamentalmente) a ordenação efectuada na base de dados de raios, algumas variações podem no entanto ser definidas [Sou92]:

- Adaptativa
- Sequencial
- Aleatória
- Resultante de uma divisão recursiva

Por ordem adaptativa, entende-se que as amostras são calculadas e, consequentemente as células imagem são actualizadas, tendo em atenção as zonas da imagem com maior

detalhe (exploração da coerência da imagem). Neste caso, um factor de homogeneidade é necessário para classificar cada célula, e a prioridade desta é influenciada por ele de forma que algumas células de menores dimensões possam ser actualizadas antes de outras maiores mas mais homogéneas.

Numa ordem sequencial, células que possuam dimensões idênticas são actualizadas segundo o processo de varrimento de uma imagem (de cima para baixo e da esquerda para a direita). A ordem aleatória é óbvia.

Se as células imagem vão sendo geradas à medida que são processadas, uma estratégia de subdivisão recursiva pode ser definida. A ordem pela qual são actualizadas é afectada de acordo.

Várias experiências conduzidas no sentido de avaliar os resultados das possíveis combinações de tipos de células com diferentes ordens de actualização mostram que:

1. As células imagem rectangulares com iluminação suavizada são computacionalmente muito pesadas para o sistema, pelo que não são aplicáveis. Realmente, a interpolação bilinear necessária, atendendo à arquitectura do sistema, só pode ser realizada no processador atribuído ao processo de visualização. Não existindo meios rápidos (*hardware*) para a sua realização, os tempos consumidos as actualizações resultam demasiado longos para um recurso (processador gráfico) que é único para toda a arquitectura.

2. As células imagem uniponto têm um efeito no crescimento do realismo muito suave no tempo o que torna a sua utilização bastante agradável. No entanto, dado que em toda a área de uma célula somente um ponto é iluminado, as primeiras imagens obtidas, com células de maiores dimensões, resultam tendencialmente escuras, o que dificulta a sua avaliação visual.

3. A ordem adaptativa de actualização das células imagem é difícil de implementar face à arquitectura utilizada. Note-se que a unidade mínima de cálculo é a amostra (ponto no ecrã), não existindo, na estrutura do sistema, qualquer relação entre amostras vizinhas que permita classificar uma área ecrã quanto à sua homogeneidade.

4. A ordem sequencial dá um aspecto demasiado regular (matricial) ao aparecimento ou actualização das células imagem, o que a torna bastante desagradável de observar.

5. A ordem resultante de uma divisão recursiva também se torna bastante desagradável. As células são actualizadas segundo uma ordem demasiado fractal que não se relaciona com os detalhes da imagem.

De acordo com estes resultados experimentais e por eliminação de hipóteses, restam as células imagem com iluminação constante segundo uma ordem aleatória de actualização.

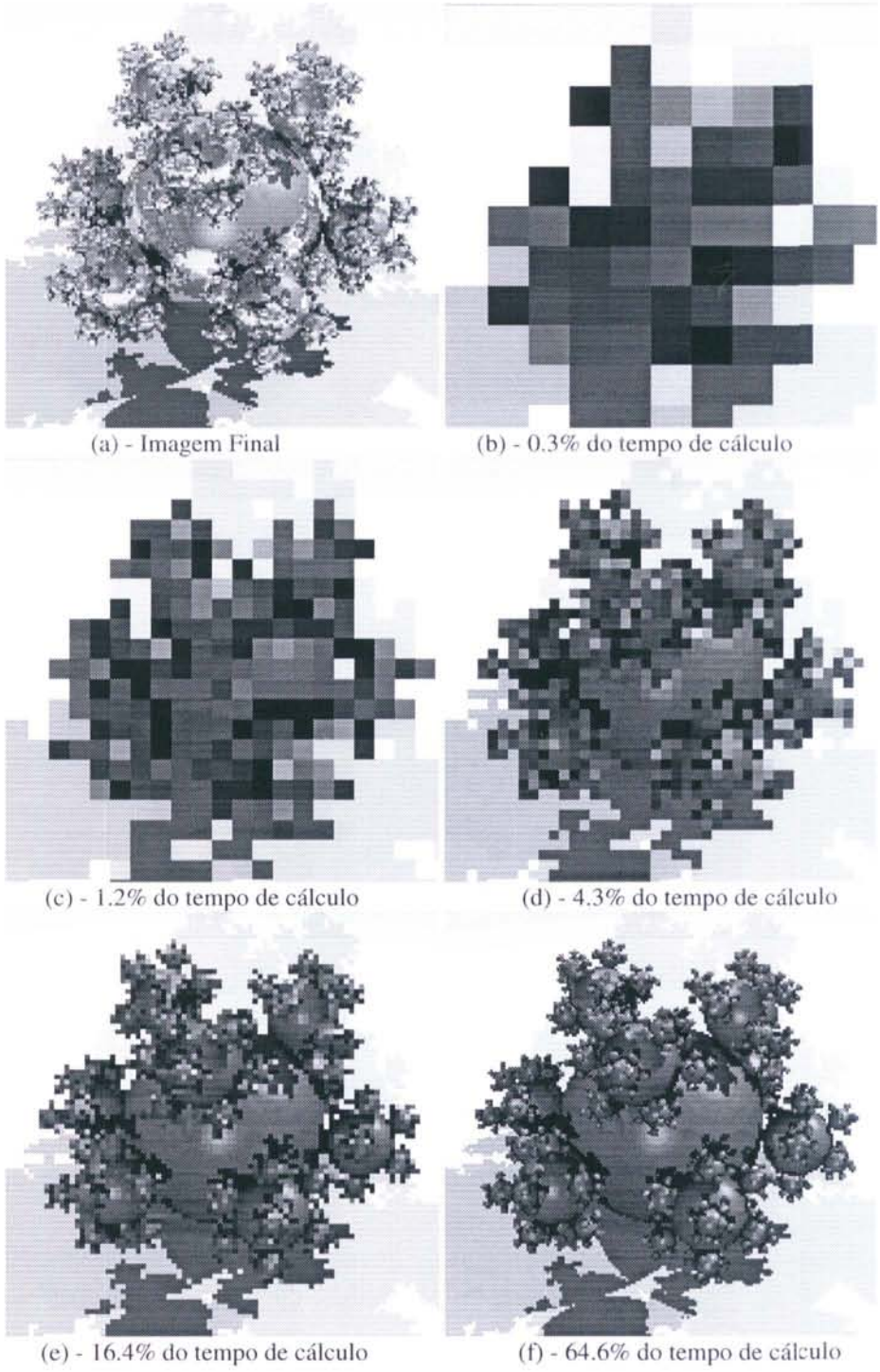
O facto de as células possuírem iluminação constante, dá às primeiras imagens, cujas células têm grandes dimensões, um aspecto de mosaico bastante acentuado. Por outro lado, a aleatoriedade da ordem de actualização das células, a uma cadência de várias centenas de células por segundo, dá um aspecto de continuidade à (progressão em) resolução que não se consegue com nenhuma das outras ordens possíveis.

O conjunto das duas soluções dá origem a imagens cujo progresso no tempo é bastante ruidoso até se atingirem células com dimensões na ordem de 4x4 ou 2x2 *pixels*. O ruído da imagem, embora um pouco distractivo é, a partir deste ponto, melhor tolerado pela vista humana do que o efeito de escada que entretanto permanece [Ama87a], [Wat92].

Nesta implementação, todos os raios iniciais (a partir do observador, com um raio por *pixel*) são criados *à priori* e cada raio corresponde ao canto superior esquerdo da célula imagem respectiva. Com a dimensão máxima das células e a partir do canto superior esquerdo do ecrã, distribuem-se as células de tamanho máximo ao longo do ecrã, gerando-se um raio por cada célula. Cada célula é depois dividida em quatro novas células, das quais, à superior esquerda não é atribuído qualquer raio (partilha do raio entre células mãe e filha).

Este processo repete-se para cada uma das novas células até se atingir a dimensão de 1x1 *pixels* e as células (de iguais dimensões) são distribuídas aleatoriamente.

A figura 4.18(a) é uma imagem obtida pelo sistema, com qualidade final. O tempo necessário para o seu cálculo é de 129 segundos. Na mesma figura, de (b) a (f), apresenta-se uma sucessão de imagens obtidas com o sistema, em realismo crescente, por progressão na resolução, começando com células imagem de 16x16 *pixels*, 8x8, etc., até 1x1. As árvores de iluminação são mantidas no primeiro nível. O esforço computacional para atingir cada uma das imagens é apresentado junto da imagem respectiva na forma percentual do tempo de cálculo da imagem final.



(a) - Imagem Final (b) - 0.3% do tempo de cálculo

(c) - 1.2% do tempo de cálculo (d) - 4.3% do tempo de cálculo

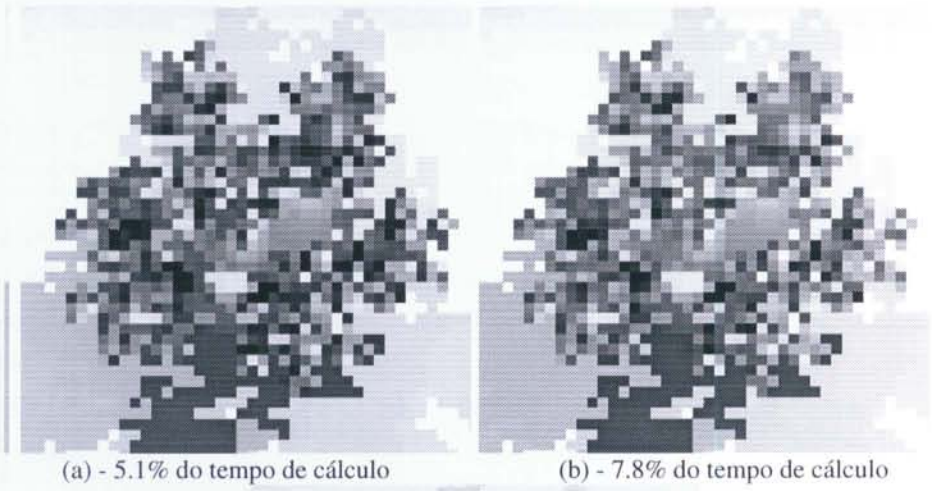
(e) - 16.4% do tempo de cálculo (f) - 64.6% do tempo de cálculo

Fig. 4.18 (Original na Secção a Cores) - Imagem final e uma sequência de realismo crescente por progressão em resolução

Uma primeira constatação, prende-se com o esforço computacional relativo à imagem que, em qualidade, se aproxima mais da imagem final, ou seja, a imagem (f). Realmente,

o tempo consumido para se atingir este nível de realismo é demasiado elevado (64% do tempo total).

É importante salientar que, o utilizador, normalmente um *designer* treinado, possui uma imagem mental já formada sobre o modelo. Nestas condições, as imagens (d) e (e) possuem algum significado, com um custo computacional bastante baixo.



(a) - 5.1% do tempo de cálculo (b) - 7.8% do tempo de cálculo

(c) - 23.5% do tempo de cálculo

Fig. 4.19 (Original na Secção a Cores) - Sequência de realismo crescente por progressão em profundidade na árvore de iluminação e em resolução

Uma nova sequência é apresentada na figura 4.19. Na imagem (a), as células imagem possuem dimensões 4x4 *pixels* e a profundidade da árvore de iluminação é 2. Nas duas seguintes, (b) e (c), a profundidade da árvore de iluminação é ilimitada[1] e as dimensões

[1] O termo ilimitada refere-se ao parâmetro de controlo de realismo crescente respectivo. Sob o ponto de vista de *ray-tracing*, a profundidade da árvore de iluminação é sempre limitada a um valor máximo de forma a evitar ciclos infinitos.

das células são respectivamente, 4x4 e 2x2 *pixels*. Com esta sequência pretende-se mostrar alguns feitos ópticos, com uma resolução baixa. Para um utilizador treinado, mesmo a imagem (a) poderá ter informação visual suficiente para uma avaliação grosseira, permitindo-lhe, eventualmente, rejeitá-la se não corresponde minimamente ao que pretende.

Suponha-se que o utilizador necessita de visualizar um determinado detalhe. A hipótese de definir uma área de interesse é, neste caso um precioso auxiliar. Um exemplo é apresentado na figura 4.20.

Em conclusão, o sistema de *ray-tracing* com realismo crescente apresentado, permite ao utilizador acompanhar a criação da imagem desde as aproximações mais grosseiras até à imagem final. A qualquer momento e com a informação visual disponibilizada, o utilizador pode decidir que, eventualmente, a imagem não está a corresponder ao que pretende, e portanto, o cálculo pode ser abortado num estágio prematuro.

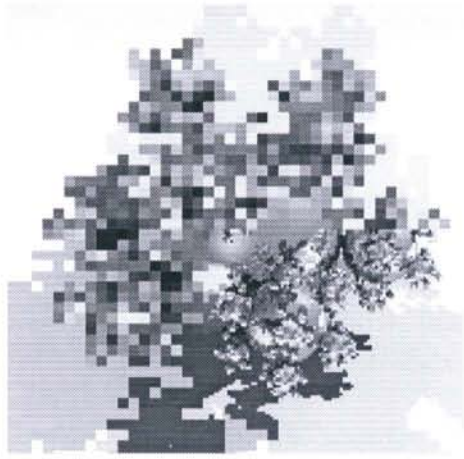


Fig. 4.20 (Original na Secção a Cores) - Definição de uma área de interesse

Os principais problemas do método são, o acentuado efeito de mosaico das imagens parciais e o facto de o sistema não utilizar informação de coerência da imagem para, automaticamente, concentrar esforço computacional nas áreas de maior detalhe. Por exemplo, as células situadas na periferia da imagem (onde é projectado o chão) são bastante coerentes, mas são subdivididas exactamente da mesma forma que as células imagem que contêm as mais pequenas esferas. O tempo de cálculo consumido com as primeiras, se aplicado nas segundas, permitiria obter sensivelmente as mesmas imagens em menores porções de tempo.

4.3.8 Avaliação da Paralelização

Para efeitos de avaliação do sistema, considera-se a sua arquitectura composta de três processadores fixos para os processos Visualização, Interacção/Coordenação e Base de Dados de Raios e de um número variável de processadores, até um máximo de sete[1], para o processo Cálculo de Intersecções. Pretende-se, com esta avaliação, determinar a evolução do comportamento do sistema com o aumento do número de *workers* que constitui o processo Cálculo de Intersecções.

A topologia da rede de *workers* tem como base uma árvore. Outras topologias experimentadas no âmbito do trabalho apresentam piores resultados, o que se justifica pelo facto de a troca de mensagens se efectuar sempre entre um qualquer *worker* e o *master* e nunca entre dois *workers*. Com a limitação física de cada processador (*transputer*) possuir quatro ligações físicas com o exterior (*links*), a árvore de maior grau possível é a ternária (figura 2.8).

À medida que o número de processadores aumenta, estes distribuem-se uniformemente pelos ramos da árvore, de forma a que esta resulte o mais equilibrada possível. Note-se que os processadores mais próximos da raiz são atravessados por uma grande quantidade de mensagens cujo destino ou proveniência é um outro processador que não o próprio. Admitindo que um processador tem que dedicar algum tempo de CPU (ainda que pequeno) à gestão dessas mensagens e que esse tempo é perdido em termos de cálculo de intersecções, torna-se importante distribuir os processadores disponíveis pelos ramos da árvore, equilibrando-a em termos físicos e, em consequência, em termos de disponibilização de capacidade de processamento.

A figura 4.21 mostra a rede utilizada de *workers* desde um processador até ao máximo de sete, respeitando, em cada situação, o equilíbrio físico da árvore.

[1] Limitação decorrente do total de processadores disponíveis.

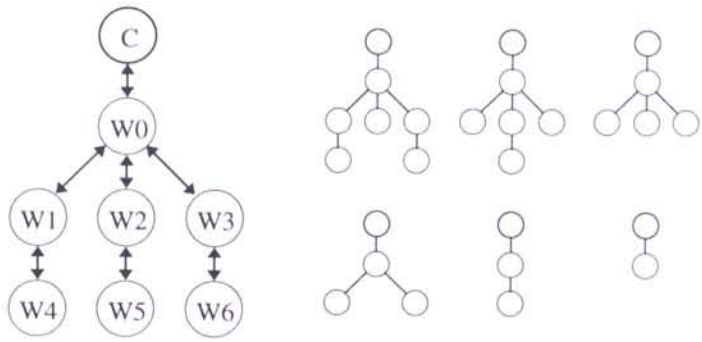


Fig. 4.21 - As sete topologias utilizadas na avaliação da paralelização

A avaliação da paralelização do sistema concentra-se na medida dos tempos correspondentes à síntese de uma imagem completa em modo interactivo, com diferentes quantidades de *workers* e na determinação dos respectivos *speedup* e eficiência.

A base utilizada como cena de teste é retirada de *Standard Procedural Database* (SPD) [Hai87]. Compõe-se de esferas fortemente especulares, sobre um chão de características difusas e possui três fontes de luz pontuais. Com esta base foram criadas várias cenas de teste, com complexidades diferentes, fazendo variar o número de esferas entre 200 (considerada uma cena simples) e 3200 (cena medianamente complexa). Durante a geração de cada imagem, contabiliza-se a quantidade de raios intersectados, de forma a permitir a avaliação do tempo de processamento por raio, necessária para perceber a evolução do comportamento do sistema com a complexidade da cena.

No quadro 4.2 apresentam-se os tempos de cálculo medidos na obtenção de uma imagem completa, para as várias cenas de teste, com um número crescente de processadores (*workers*).

	Nº de Workers						
Nº Objs.	1	2	3	4	5	6	7
200	145.9	75.0	50.8	40.3	35.9	34.3	33.7
400	189.3	96.9	65.3	49.8	41.8	38.3	36.9
800	218.4	111.4	75.0	56.9	46.4	40.7	38.3
1600	272.7	138.6	93.2	70.4	56.8	48.0	42.5
3200	323.2	164.5	110.0	83.0	66.9	56.2	48.8

Quadro 4.2 - Tempos de cálculo, em segundos, de imagens de complexidades várias e com diferentes números de *workers*

A determinação do *speedup* (quadro 4.3) é efectuada tendo como base o tempo consumido pela arquitectura com um *worker*, ou seja, equivale ao termo *scaled speedup* de J. Gustafson [Gus88] e referido na secção [2.3.1]. Esta opção justifica-se por

imposição do próprio sistema, dado que os processos Visualização e Base de Dados de Raios são implementados em processadores dedicados, impossibilitando a versão uniprocessador.

	Nº de Workers						
Nº Objs.	1	2	3	4	5	6	7
200	1.00	1.95	2.87	3.62	4.06	4.25	4.33
400	1.00	1.95	2.90	3.80	4.53	4.94	5.13
800	1.00	1.96	2.91	3.84	4.71	5.37	5.70
1600	1.00	1.97	2.93	3.87	4.80	5.68	6.42
3200	1.00	1.96	2.94	3.89	4.83	5.75	6.62
S. Ideal	1.00	2.00	3.00	4.00	5.00	6.00	7.00

Quadro 4.3 - *Speedups* obtidos para imagens de complexidades várias

No quadro 4.4 apresenta-se a eficiência calculada com um número crescente de *workers* e para as várias cenas de teste. Nota-se desde já uma diminuição da eficiência do sistema, principalmente para as cenas mais simples e com um maior número de *workers*.

	Nº de Workers						
Nº Objs.	1	2	3	4	5	6	7
200	100.0%	97.3%	95.7%	90.5%	81.3%	70.9%	61.8%
400	100.0%	97.7%	96.6%	95.0%	90.6%	82.4%	73.3%
800	100.0%	98.0%	97.1%	96.0%	94.1%	89.4%	81.5%
1600	100.0%	98.4%	97.5%	96.8%	96.0%	94.7%	91.7%
3200	100.0%	98.2%	97.9%	97.3%	96.6%	95.8%	94.6%

Quadro 4.4 - Eficiências obtidas para imagens de complexidades várias

As figuras 4.22, 4.23 e 4.24 resumem em forma gráfica o conteúdo dos quadros anteriores, pela mesma ordem.

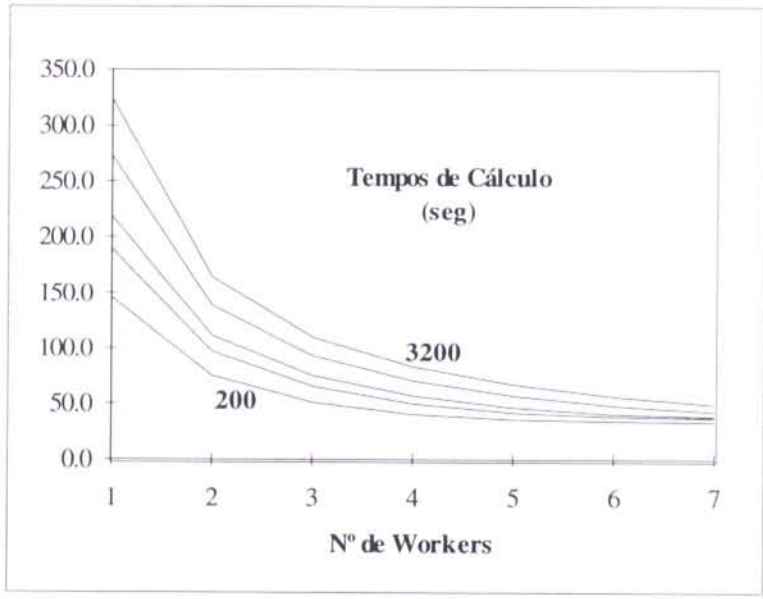


Fig. 4.22 - Tempos de cálculo de várias imagens de complexidades várias e com diferentes números de *workers*

Na figura 4.23 apresentam-se as várias curvas de *speedup*, correspondentes às diferentes cenas de teste. Acrescenta-se ainda a linha recta correspondente ao *speedup* ideal (recta com inclinação unitária) de modo a permitir uma melhor avaliação do *speedup*.

É visível que o sistema apresenta um comportamento razoavelmente próximo do ideal nas cenas de maior complexidade. No entanto, as curvas das cenas mais simples mostram a tendência do sistema para entrar em saturação quando o número de *workers* ultrapassa determinado valor que depende da complexidade da cena.

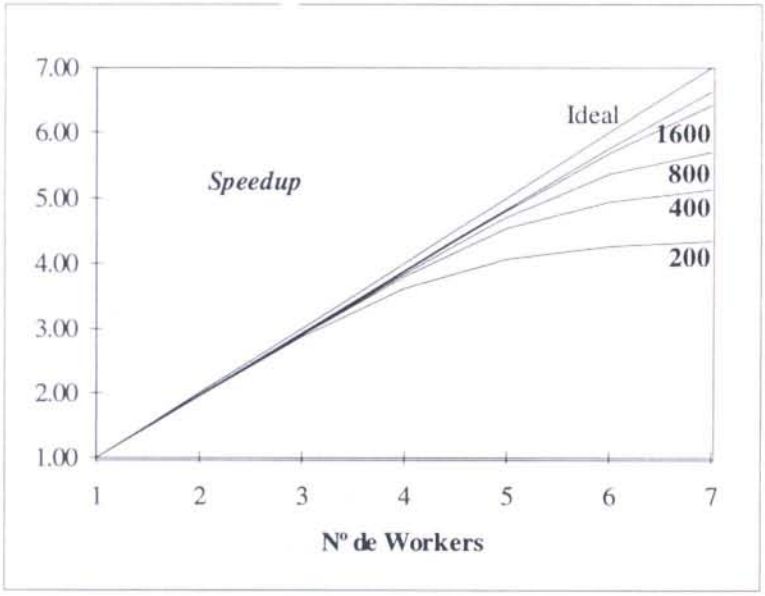


Fig. 4.23 - *Speedups* obtidos para imagens de complexidades várias

Tomando como exemplo a curva de eficiência (figura 4.24) correspondente à cena mais simples, pode verificar-se que, com quatro processadores, a eficiência do sistema ainda se mantém num valor bastante aceitável de cerca de 90%. No entanto, com cinco processadores, a eficiência baixa para cerca de 80% e continua a baixar de uma forma quase linear com o aumento do número de processadores. A curva respectiva na figura 4.23 mostra claramente que, acima dos cinco ou seis processadores, o *speedup* já praticamente não aumenta, o que denota a saturação do sistema ou, dito de outra forma, o sistema, com mais de quatro *workers*, encontra-se sobredimensionado para o problema em causa.

Repare-se entretanto que o ponto de saturação se afasta para cima e para a direita à medida que a complexidade do problema aumenta. Esta observação está de acordo com o efeito de Amdahl referido na secção [2.3.1].

De acordo com esta tendência, é interessante verificar que, com os dados disponíveis, o sistema tem um comportamento próximo do princípio de Gustafson. Segundo este princípio, o *speedup* S_{up} de um sistema paralelo com N_p processadores relaciona-se com a componente puramente sequencial s através da equação 2.3 de onde:

$$s = \frac{N_p - S_{up}}{N_p - 1} \qquad \text{Eq. 4.29}$$

Os quadros 4.5 e 4.6 mostram a evolução da componente sequencial *s* quando a complexidade do problema aumenta na mesma proporção do número de processadores (na impossibilidade de obter dados com oito processadores, apresenta-se, como aproximação, o valor de *s* obtido com sete, o que faz aumentar um pouco o valor de *s*). A componente sequencial denota uma tendência a aumentar ligeiramente, mas na ordem de 5% no quadro 4.5 e de 4% no quadro 4.6.

Nº Work.s	**1**	**2**	**4**	**7**
Nº Objs.	**200**	**400**	**800**	**1600**
Speedup	**1.00**	**1.95**	**3.84**	**6.42**
S	-	4.6%	5.4%	9.7%

Quadro 4.5 - Evolução da componente sequencial do sistema com o aumento da complexidade do problema e do número de processadores (I)

Em resumo, o sistema tem um comportamento que se pode considerar bom se utilizado para sintetizar imagens de cenas complexas. Quando as cenas são simples, a componente sequencial introduzida tende a aumentar percentualmente e o *speedup* afasta-se da linha ideal.

Nº Work.s	**1**	**2**	**4**	**7**
Nº Objs.	**400**	**800**	**1600**	**3200**
Speedup	**1.00**	**1.96**	**3.87**	**6.62**
S	-	3.9%	4.2%	6.3%

Quadro 4.6 - Evolução da componente sequencial do sistema com o aumento da complexidade do problema e do número de processadores (II)

As mesmas conclusões podem ser retiradas da figura 4.24. As cenas com 1600 e 3200 objectos mantêm uma eficiência superior a 90%, mesmo com todos os sete *workers* activos. Na cena mais simples, com 200 objectos, a eficiência diminui muito cedo (aos quatro processadores inclusive), tornando o sistema sobredimensionado para o problema em causa.

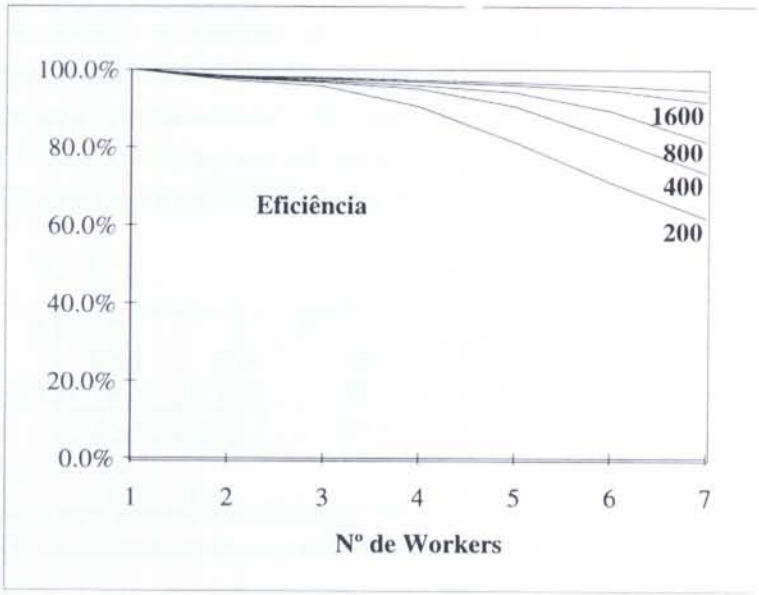


Fig. 4.24 - Eficiências obtidas para imagens de complexidades várias

Uma faaceta interessante do sistema a estudar é a evolução dos tempos de cálculo com a complexxidade da cena. No entanto, cenas diferentes, ainda que com a mesma resolução, requererm um total de raios a intersectar também muito diferentes, pelo que os tempos totais dde cálculo não são comparáveis. Uma normalização baseada na avaliação do tempo médio dde processamento por raio é portanto necessária.

O quaddro 4.7 resume os tempos médios de processamento por raio, para diferentes númerosss de *workers* e diferentes complexidades da cena.

	Nº de Workers						
Nº Objs.	1	2	3	4	5	6	7
200	2.640	1.357	0.919	0.729	0.650	0.621	0.610
400	3.225	1.651	1.113	0.849	0.712	0.653	0.629
800	3.672	1.873	1.261	0.957	0.780	0.684	0.644
1600	4.470	2.272	1.528	1.154	0.931	0.787	0.697
3200	5.087	2.589	1.731	1.306	1.053	0.884	0.768

QQuadro 4.7 - Tempos de cálculo por raio, em milisegundos, nas várias cenas

A sua innterpretação, feita por linha, é a mesma que a do quadro 4.2 e portanto resulta num grááfico semelhante ao da figura 4.22. Os valores apresentados são calculados pelo quocientte do tempo total de síntese de uma imagem pelo total de raios processados, incluinddo, por conseguinte, parcelas de cálculo e de transmissão de mensagens.

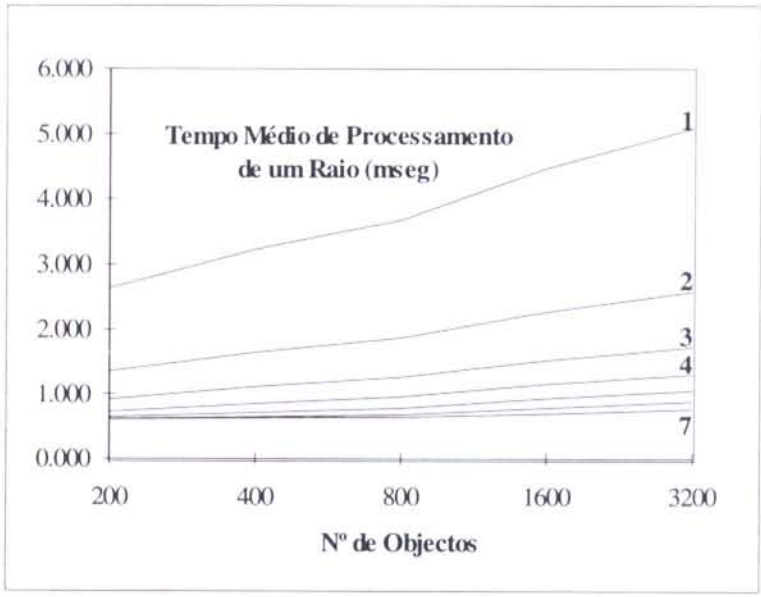


Fig. 4.25 - Tempos médios de cálculo por raio, nas várias cenas

Como ponto de partida, seja o sistema com um único *worker* que apresenta uma eficiência de 100%, isto é, sem tempos de inactividade. A evolução dos tempos por raio apresentada na figura 4.25 sugere uma evolução logarítmica em função do número de objectos N_{Ob}:

$$t = t_0 + k_t \cdot \log(N_{Ob}) \qquad \text{Eq. 4.30}$$

Realmente, utilizando a cena com 200 objectos para a determinação de t_0, verifica-se que o valor de k_t que satisfaz a equação 4.30 nas outras cenas (com um *worker*, recorde-se) é sensivelmente constante (aproximadamente 2, quadro 4.8).

	Nº de Workers						
Nº Objs.	1	2	3	4	5	6	7
200	-	-	-	-	-	-	-
400	1.94	0.98	0.64	0.40	0.21	0.11	0.06
800	1.71	0.86	0.57	0.38	0.22	0.11	0.06
1600	2.03	1.01	0.67	0.47	0.31	0.18	0.10
3200	2.03	1.02	0.67	0.48	0.33	0.22	0.13

Quadro 4.8 - Coeficientes k_t da função de tempo de processamento por raio.

O mesmo processo de cálculo dos coeficientes k_t aplicado aos restantes casos (variação com o número de *workers*) conduz aos resultados da figura 4.26.

O comportamento logarítmico dos tempos por raio é bastante razoável em qualquer sistema de *ray-tracing*, principalmente se se pretende utilizar em situações extremas de grande complexidade das cenas. No entanto, conforme as curvas inferiores da figura 4.26 deixam perceber, os coeficientes k_t das arquitecturas com maior número de *workers* tendem a crescer o que, mais uma vez, se deve ao abaixamento da eficiência do sistema nestas arquitecturas.

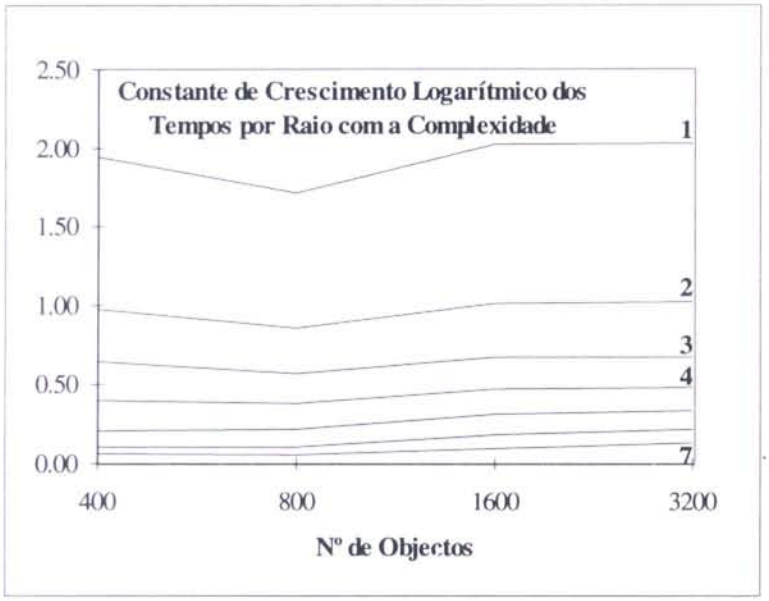


Fig. 4.26 - Coeficientes k_t da função de tempo de processamento por raio

Em resumo, com o número de processadores disponíveis, a paralelização do sistema apresenta resultados de *speedup* e de eficiência próximos do ideal desde que a cena apresente um grau de complexidade não muito baixo.

De acordo com os quadros 4.4 e 4.7, a eficiência mantém-se na casa dos 90% desde que o conjunto de *workers* disponíveis processem raios a uma cadência não superior a 1/0.7mseg. Obviamente, a linearidade do sistema observada nos casos de teste com cenas mais complexas desapareceria se o número de processadores continuasse a aumentar. As razões de tal comportamento podem ser de natureza arquitectural ou tecnológica, esta última relacionando-se com a eventual saturação das linhas de comunicação de mensagens entre processadores.

A comunicação de mensagens em arquitecturas baseadas em *transputers* faz-se através das linhas série designadas por *links* cuja taxa máxima de comunicação unidireccional é, para pacotes de informação semelhantes aos de um raio, 1.3M*Bytes*/seg. Dentro de um ciclo de 0.7mseg (cadência referida para os *workers*), são comunicadas, na situação mais

desfavorável, cinco mensagens de raios com 80 *Bytes* cada uma, através do *link* mais desfavorecido (o que liga os processos Coordenação e Base de Dados de Raios). A esta situação corresponde uma taxa de comunicação de 0.57M*Bytes*/seg, inferior à taxa máxima referida e portanto a diminuição da eficiência do sistema não pode ser atribuída à saturação das linhas de comunicação.

O problema parece ser portanto de natureza arquitectural. Realmente, de acordo com as teorias de Amdahl e Gustafson, o sistema apresenta uma componente sequencial apreciável. Dado que a paralelização ao nível dos *workers* é bastante eficiente[1], então a componente sequencial é oriunda do conjunto Coordenação/Base de Dados de Raios/Visualização.

Efectivamente, o conjunto de operações desencadeado nestes processos pela chegada de um raio intersectado vindo do processo Cálculo de Intersecções é grande e difícil de paralelizar. Em resumo, a granularidade das tarefas a executar pelos *workers* é bastante fina, mas a granularidade das tarefas dos processos centrais é bastante grossa, as duas não se coordenando mutuamente no tempo quando o número de processadores aumenta.

4.4 Sumário

A síntese de imagens com elevado nível de realismo é fortemente consumidora de tempo de processamento. É portanto difícil a sua compatibilização com ambientes interactivos.

Neste capítulo, resumiram-se dois trabalhos publicados que fazem evoluir a qualidade da imagem ao longo do tempo de processamento, através de uma sequência de métodos diferentes de visualização. Com essa possibilidade o utilizador pode, em pouco tempo, prever toscamente o resultado final e decidir se a imagem corresponde ao esperado, permitindo ou não a continuidade do seu processamento até à qualidade final desejada.

A utilização de métodos diferentes não é a solução ideal pois pode conduzir a saltos demasiado bruscos na qualidade da imagem e até mesmo a outros problemas de eventual inadaptação dos métodos. Por outro lado, o algoritmo *ray-tracing* tradicional, cuja qualidade das imagens é bastante boa, não permite a observação de estágios parciais das mesmas.

Uma nova definição de *ray-tracing* foi apresentada, com nível crescente de realismo, segundo a qual uma imagem não é calculada definitivamente com a qualidade final

[1] Embora não se tenha referido, a presença de monitores nos *workers* permite saber que a taxa de ocupação de CPU em cada um deles é, na fase linear do sistema, sempre superior a 95%.

desejada, mas sim com uma qualidade que é progressivamente maior com o tempo de processamento. Para o efeito, nomearam-se as características do algoritmo tradicional que influenciam a qualidade da imagem e definiram-se os respectivos parâmetros de controlo. Estes são utilizados para limitar o nível de realismo que a dado momento é pretendido.

Duas implementações experimentais, desenvolvidas localmente, foram descritas, uma sequencial e outra sobre uma arquitectura paralela.

A versão sequencial serviu para validar a noção de *ray-tracing* com nível crescente de realismo. Apresenta dois problemas associados que são, a necessidade de repetição (embora limitada) de cálculos já efectuados e a pouca interactividade no controlo do realismo da imagem, dado que funciona por passos.

A versão sobre uma arquitectura paralela surge pela necessidade de, apesar das facilidades introduzidas pela noção de realismo crescente, ser necessário acelerar os cálculos, intrinsecamente demorados, de intersecções de raios com objectos.

As avaliações efectuadas a esta última versão, do ponto de vista de realismo crescente, mostram a validade do método e algumas das suas carências, nomeadamente o acentuado efeito de mosaico que é observado nas primeiras versões da imagem, quando a resolução é menor.

Do ponto de vista de paralelização, o sistema, dotado de um pequeno número de processadores, tem um comportamento próximo do ideal quando as cenas a processar são complexas. No entanto, o processamento de cenas simples denuncia a saturação da capacidade de processamento dos processos não paralelizáveis que são de serviço comum (Coordenação, Base de Dados de Raios e Visualização).

A observação das curvas de tempos de cálculo das várias imagens de teste (figura 4.22) mostram que, com um número crescente de processadores, o tempo necessário à obtenção de uma imagem tende para um valor constante (na ordem dos 20 a 30 mseg nos casos observados, em que a resolução da imagem é 100x100 *pixels*). Este é o tempo mínimo conseguido com o sistema, qualquer que seja o número de processadores disponíveis.

Consoante a aplicação, o tempo mínimo conseguido pode não ser compatível com a interactividade necessária. Por outro lado e de acordo com a Lei de Amdahl, haverá sempre um limiar de processadores a partir do qual o tempo de cálculo de uma imagem não se consegue diminuir, por melhor paralelizado que seja o sistema.

Torna-se portanto importante redefinir a paralelização do sistema de forma a aumentar a sua eficiência e, em consequência, diminuir o tempo mínimo de síntese de uma imagem e aumentar o limiar de processadores referido.

Capítulo 5

IIRRA: Algoritmo Interactivo de *Ray-Tracing* com Nível Crescente de Realismo

5. IIRRA: ALGORITMO INTERACTIVO DE RAY-TRACING COM NÍVEL CRESCENTE DE REALISMO 5.1

5. IIRRA: Algoritmo Interactivo de *Ray-Tracing* com Nível Crescente de Realismo

No capítulo 4 descreveu-se a noção de realismo crescente e apresentaram-se duas implementações experimentais.

A primeira implementação, obtida por alteração de um sistema de *ray-tracing* sequencial existente, é destinada a avaliar as possibilidades das ideias em embrião [4.1]. A segunda, desenvolvida de raiz para um sistema em arquitectura paralela baseada em *transputers*, é mais próxima de uma solução final para o problema [4.2].

A implementação experimental em arquitectura paralela possui vantagens sobre a primeira no que respeita à interactividade permitida durante a síntese de uma imagem com qualidade progressiva. Apresenta no entanto alguns problemas, tanto na visualização com realismo crescente, como na paralelização.

Neste capítulo descrevem-se os melhoramentos julgados necessários para a definição de um novo sistema de síntese de imagem, designado por IIRRA (*Interactive Increasing Realism Ray-tracing Algorithm*).

As experiências efectuadas para a avaliação do realismo crescente no sistema em arquitectura paralela da secção [4.2] mostram que, a progressão em profundidade nas árvores de iluminação, assim como a definição de áreas de interesse, se encontram correctamente estabelecidas, sendo o seu efeito o esperado. No entanto, a progressão em

resolução apresenta basicamente dois problemas que interessa salientar e que são o forte efeito de mosaico e a impossibilidade de concentra o processamento nas células de maior detalhe. Ambos os problemas têm origem na reduzida informação disponível por cada célula imagem, somente uma amostra, e na total independência observada entre as várias amostras.

No que respeita à paralelização, o balanceamento de carga computacional conseguido não é suficiente para garantir a eficiência do sistema. A diminuição desta é notada principalmente quando as cenas são demasiado simples ou quando o número de processadores é elevado e deve-se a uma granularidade de tarefas muito fina, em oposição à granularidade grossa do conjunto de processos de serviços comuns. Globalmente, pode afirmar-se que o centro do problema se situa ao nível do método de paralelização utilizado, em *farm* de processadores, que centraliza a gestão de tarefas num único conjunto de processos Coordenação/Visualização/Base de Dados de Raios. Quando a taxa de tarefas processadas aumenta, este conjunto fica sobrecarregado, tanto ao nível de comunicações, como ao nível do processamento, não conseguindo acompanhar os restantes processos do sistema.

Um avanço no sentido de solucionar o problema da eficiência passa pela adopção de um maior peso computacional de cada tarefa, devendo esta passar a definir um conjunto de amostras que, de alguma forma, se relacionam entre si[1]. Este facto, conjugado com a necessidade de aumentar a informação disponível por cada célula imagem, sugere que uma tarefa seja definida como uma célula imagem caracterizada por várias amostras. O crescimento do realismo ganha com esta solução, pois a maior informação disponível por célula possibilita o aumento das suas dimensões e, simultaneamente, a redução do efeito de mosaico.

Por outro lado, a gestão de tarefas deve ser descentralizada, passando a ser efectuada igualmente por cada um dos processadores disponíveis. Cada processador deve passar a gerir as tarefas de que é incumbido, interagindo, eventualmente, com outros processadores, de forma a permitir uma visão global da gestão de tarefas.

A descentralização, além de aumentar a quantidade de processamento paralelizado, diminui as comunicações, dado que as tarefas deslocadas entre um qualquer processador e o de coordenação passa a ser muito inferior.

Do ponto de vista das teorias de Amdahl [Amd67] e de Gustafson [Gus88], a descentralização tem o efeito de diminuir as componentes sequenciais do sistema, melhorando consequentemente a eficiência do mesmo [2.3].

[1] Na solução anterior, uma tarefa corresponde ao cálculo de intersecções de um raio com objectos.

Também os dados relativos a entidades acedidas por vários processadores (objectos, superfícies, fontes de luz) não devem ser residentes em memória central. Este aspecto é também um dos pontos fracos da solução em arquitectura paralela apresentada na secção [4.2], pois a replicação da base de dados de objectos em todos os *workers* impede, por motivos de quantidade de memória disponível por processador, a síntese de imagens de cenas com grande complexidade.

5.1 Melhoria da Eficiência da Paralelização

O problema de acesso a dados de entidades comuns a vários processadores é conhecido. Soluções do tipo das utilizadas nos sistemas de gestão de memória virtual são propostas em trabalhos publicados sobre paralelização de *ray-tracing* [Gre89], [Gre91], [Bad90], [Bad90a], [Bad94]. Na origem dessas soluções está a manutenção, na memória local de cada processador, dos dados que, estatisticamente, têm vindo a ser mais solicitados nos últimos acessos. Sempre que aconteça um acesso a um determinado item de dados que não se encontre presente localmente, um pedido é enviado para a rede de processadores no sentido de preencher o item em falta.

A implementação do sistema em arquitectura paralela secção [4.2] é realizada segundo uma *farm* de processadores [4.2.8.1], pelo que a rede de comunicações se encontra dividida em duas grandes vias, a dos dados e a dos resultados. Realmente, as mensagens daquele sistema são quase exclusivamente de dois tipos: mensagens enviadas do *master* para os *workers*, contendo dados/tarefas e mensagens enviadas dos *workers* para o *master*, contendo os resultados das tarefas executadas. Não existem mensagens de *worker* para *worker*.

O tipo de soluções mencionado para a acessibilidade aos dados expande a utilização da rede de comunicações entre processadores, dado que passam a existir mensagens inter-*workers*. S. Green, em [Gre91], propõe, para o seu sistema de *ray-tracing* (DEnIS-*Distributed Environment for Information Sharing*), uma arquitectura com acesso a uma rede global de processadores que se representa na figura 5.1.

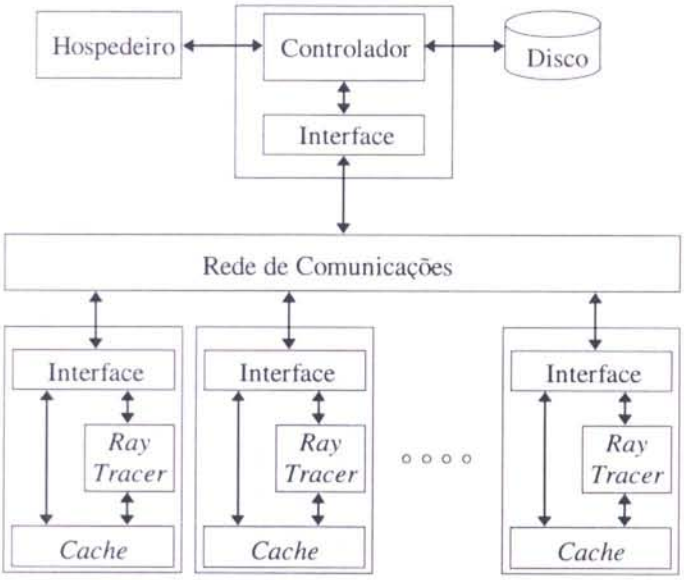


Fig. 5.1 - Arquitectura do sistema DEnIS, de S. Green

O problema de descentralização da gestão de tarefas é resolvido, no sistema DEnIS, com a colocação, em cada *worker*, de um processo mais ou menos autónomo, que efectua essa gestão e que envia pedidos para a rede sempre que o conjunto de tarefas que contém seja inferior a um determinado limiar.

No sistema DEnIS, existem três processos em cada *worker* (figura 5.2):

WP *Work Process*: processo que executa as tarefas;

WDM *Work Distribution Manager*: comunica com a rede e gere tarefas, atribuindo-as ao processo WP local ou partilhando-as com um WDM de outro processador;

DBM *Database Manager*: gere os dados de acordo com estratégias de gestão de memória virtual; à falta de um dado, envia um pedido à rede através do WDM.

De acordo com as afirmações anteriores sobre distribuição da gestão de tarefas, gestão de dados e comunicações, a composição do sistema DEnIS é aplicável no sistema IIRRA.

No entanto, o sistema DEnIS difere substancialmente do sistema IIRRA ao nível da gestão de tarefas. Embora a definição de tarefa seja semelhante nos dois sistemas (uma tarefa é uma célula imagem), a gestão das mesmas é muito mais complexa em IIRRA.

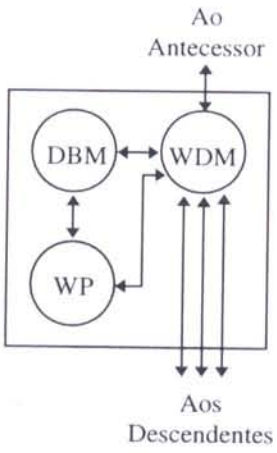


Fig. 5.2 - Os três processos de um *worker* no sistema DEnIS

DEnIS é um sistema de *ray-tracing* do tipo tradicional que sintetiza uma imagem ponto a ponto até à exaustão. Ou seja, uma vez iniciada uma tarefa (relembre-se, célula imagem), o gestor de tarefas só volta a ser requisitado quando a tarefa terminar ou seja, quando todas as suas amostras, com a resolução final, tiverem sido processadas até à profundidade máxima na árvore de iluminação. Durante o tempo que decorre entre os dois eventos, o processo WDM pode concentrar-se no atendimento de outros pedidos, quer da parte dos processos seus concorrentes no mesmo processador, quer da parte da rede.

Pelo contrário, uma tarefa, tal como é definida em IIRRA, só na fase final do processamento da imagem é dada por terminada. Realmente, a progressividade em resolução e em profundidade na árvore de iluminação tem que ser efectuada simultaneamente nas várias células imagem. De cada tarefa extraída do gestor de tarefas, só é executada uma microtarefa e, portanto, a taxa de acessos ao gestor de tarefas é, no sistema IIRRA, bastante elevada.

Em conclusão, a solução adoptada por S. Green para a composição do sistema DEnIS não é a mais adequada para o sistema IIRRA. Efectivamente, o processo WDM, demasiado ocupado com a gestão de tarefas, acabaria por ser sede de atrasos nas comunicações da rede, pelas quais é responsável em nome do processador em que se insere.

A. Chalmers, em [Cha91a], apresenta um sistema de Síntese de Imagem baseado no algoritmo radiosidade, sobre uma arquitectura paralela baseada em *transputers*. Neste sistema, cada processador é responsável por um número idêntico de *patches* e troca, com outros processadores, informação relacionada com factores de forma e radiosidades, segundo uma estrutura que designa por AMP-*Minimum Path Configuration*.

Sob o ponto de vista da paralelização, a filosofia do sistema AMP é semelhante à de IIRRA. As tarefas inerentes ao processamento a efectuar com cada *patch*, em AMP, identificam-se com as tarefas a executar com cada célula imagem do sistema IIRRA. Isto é, as tarefas mantêm-se nos respectivos processadores, que sobre elas executam microtarefas, desencadeando uma série de comunicações através da rede para efeitos de transmissão de dados.

Uma descrição mais pormenorizada do sistema AMP pode ser encontrada em [Cha91]. Daí se transcreve a figura 5.3 que representa o conjunto de processos de cada processador:

AP *Application Process*: executa os cálculos inerentes à aplicação (radiosidade);

TM *Task Manager*: gere a manutenção e a aquisição de novas tarefas para o processador local;

DM *Data Manager*: gere os requisitos de dados do processador local, acedendo automaticamente à rede sempre que necessário;

R *Router*: é responsável pela comunicação de mensagens entre qualquer um dos processos locais e os outros processadores da rede, assim como pelo encaminhamento de mensagens que receba e que não lhe sejam destinadas;

LC *Local Controller*: coordena as actividades de todos os outros processos locais.

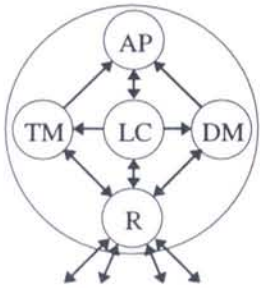


Fig. 5.3 - Os processos de um processador no sistema AMP

Face às descrições efectuadas, algumas semelhanças podem ser encontradas entre o sistema AMP e o sistema DEnIS. A principal diferença encontra-se na divisão, em processos diferentes, dos serviços de gestão de tarefas e de comunicações.

Uma boa gestão de comunicações é fundamental para a eficiência de um sistema paralelo. Esta afirmação toma especial importância no caso das arquitecturas baseadas

em *transputers*, dado que as características do seu *hardware*, se bem exploradas, optimizam a recepção e emissão de mensagens através dos canais com o exterior (*links*). Em [Atk89], apresentam-se técnicas de paralelização e de atribuição de prioridades aos processos responsáveis pelas comunicações, assim como de memorização das respectivas mensagens.

Dado que em AMP e em IIRRA, a gestão de tarefas é bastante mais consumidora de recursos computacionais do que em DEnIS, a sua separação do processo de *routing* vem facilitar a implementação das técnicas referidas em [Atk89], de modo a optimizar a gestão das comunicações.

Técnicas semelhantes de paralelização e de atribuição de prioridades aos processos responsáveis pelas comunicações são defendidas por A. d'Acierno, em [Aci92], a propósito da filosofia de programação LSGP (*Locally Sequential, Globally Parallel*).

Segundo o autor, um programa típico em linguagem OCCAM constitui-se de uma colecção de processos, vários dos quais correm em concorrência no mesmo processador. Nestas circunstâncias, algum tempo é perdido no suporte à concorrência (filas de espera de processos, mudanças de contexto) e em comunicações internas.

A filosofia LSGP aponta para uma programação paralela ao nível dos processadores, mantendo-se o código puramente sequencial em cada um deles, com vantagens óbvias nos tempos de execução[1]. No entanto e de forma a não perturbar a comunicação de mensagens inter-processadores, as porções de código relacionadas com os *routers* deverão ser sempre implementadas em concorrência.

De acordo com o esquema LSGP e com a concepção AMP de um sistema em arquitectura paralela, propõe-se, para o sistema IIRRA, a junção dos processos DM, TM e AP em um único processo, executado de um modo puramente sequencial. O diagrama correspondente à constituição do sistema proposto apresenta-se na figura 5.4.

[1] Ainda segundo o mesmo autor, as aplicações com exigências de tempo real são tipicamente desenvolvidas usando o esquema LSGP.

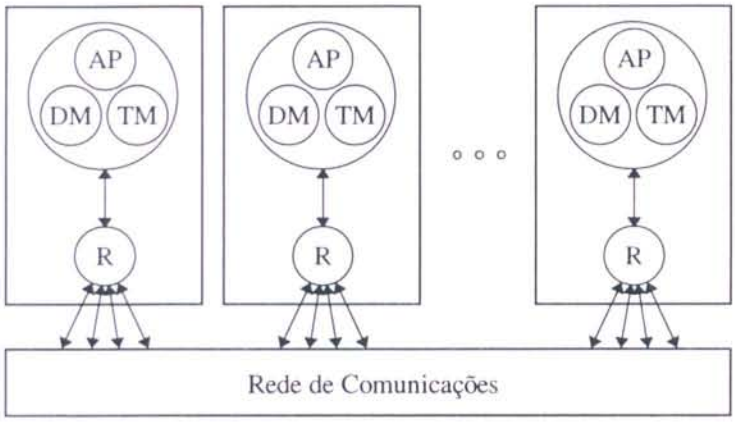


Fig. 5.4 - Arquitectura geral do sistema IIRRA

No sistema IIRRA, todos os processadores possuem uma topologia idêntica. Um deles tem funções específicas de coordenação e de serviços comuns e outro tem funções de processador gráfico. Os restantes executam réplicas do código de *ray-tracing* com realismo crescente.

Os processos de gestão de dados e de tarefas, fazendo uso da rede, podem usar estratégias de acesso remoto para adquirirem itens requeridos pelas aplicações locais e que não estejam disponíveis:

- se a localização do item é fixa num qualquer processador, este é endereçado directamente na mensagem de pedido do item em questão;
- se o item não tem localização fixa, então uma mensagem de pedido é enviada para vários processadores, começando pelos mais próximos (vizinhos) do processador que a envia.

A rede de comunicações pode ser qualquer. Devido às limitações no número de canais exteriores disponíveis por processador, uma mensagem não pode ser transmitida directamente entre dois quaisquer processadores, sendo necessário encaminhá-la por vários processadores intermédios. É da inteira responsabilidade dos *routers* encaminhar as mensagens ao longo da rede, libertando os restantes processos do conhecimento da sua topologia.

5.1.1 *Routers*

Na versão experimental em arquitectura paralela descrita na secção [4.2], o problema de *routing* é simplificado. Os dados presentes na memória local dos processadores e a forma

como as tarefas são distribuídas facilitam a implementação dos processos de *routing*, dado que as mensagens são sempre geradas entre o processo de coordenação e os *workers*.

A partilha de informação, quer ao nível dos dados, quer ao nível das tarefas, implica uma maior complexidade no desenvolvimento dos *routers* no sistema IIRRA. Genericamente, é de prever que uma mensagem possa ser enviada por um qualquer processador com destino a qualquer outro ou mesmo com destino a um grupo de processadores. Em [Sou95a] apresentam-se alguns estudos efectuados sobre *routers*, no âmbito do desenvolvimento do sistema IIRRA.

Um mesmo processador é sede de vários processos concorrentes com autonomia suficiente para enviar e receber mensagens. Assim, qualquer comunicação entre um processo (AP, DM ou TM) e a restante rede é efectuada através do *router* localizado no mesmo processador. Trata-se neste caso, de uma comunicação IE - Interior para Exterior (ou vice-versa).

Uma vez aceite pelo *router*, uma mensagem segue um determinado percurso, atravessando vários processadores até atingir o processador/processo destinatário. Em cada processador intermédio, a mensagem é recebida pelo *router* respectivo que trata de a encaminhar devidamente para outro canal. As comunicações assim efectuadas nos processadores intermédios são do tipo EE - Exterior para Exterior.

Enquanto que as comunicações do tipo EE são suportadas unicamente pelos *routers*, as comunicações do tipo IE ocorrem sempre entre um *router* e outro processo do mesmo processador. Neste contexto, o *router* é um servidor e os outros processos são clientes e por conseguinte, devem ser libertados, dentro do possível, de preocupações inerentes ao envio e à recepção de mensagens, como por exemplo o tamanho máximo das mensagens, a topologia da rede, a eventual impossibilidade momentânea de enviar mensagens e a eventualidade de ocorrência de *deadlock*.

Sintetizando, um *router* deve apresentar as seguintes características:

- Possibilitar o endereçamento a dois níveis (processador e processo);
- Suportar o endereçamento de destinatário único e outros tipos de endereçamento, de forma transparente para as aplicações;
- Tornar os percursos das mensagens transparentes aos processos clientes;
- Aplicar o mínimo de restrições às mensagens, nomeadamente ao seu comprimento;
- Evitar o *deadlock*;
- Consumir o mínimo de tempo de processamento;

- Apresentar um tempo de latência baixo (tempo que decorre entre o envio de uma mensagem e a sua recepção no destinatário).

Outras características a referir, mas que poderão não ser muito relevantes para o protótipo IIRRA, são ainda, a capacidade de auto-adaptação a qualquer topologia de rede e a tolerância às falhas.

5.1.1.1 Endereçamento de Mensagens

Em [Hwwa93], sob o título de *Communication Patterns*, classifica-se a comunicação de mensagens, quanto ao endereçamento, em quatro tipos diferentes:

Unicast:	comunicação de mensagens de uma fonte com um único destinatário;
Multicast:	comunicação de mensagens de uma fonte com vários destinatários;
Broadcast:	comunicação de mensagens de uma fonte com todos os outros como destinatários;
Conference:	comunicação de mensagens de todos para todos.

No sistema IIRRA, é previsível a necessidade de utilização dos três primeiros tipos de mensagens enunciados.

Atendendo à constituição do sistema representada na figura 5.4, na qual um processador contém vários processos independentes e com responsabilidades diferentes, é de prever que uma mensagem recebida por um processador só tenha interesse para um determinado processo. O endereçamento a dois níveis, referido anteriormente, relaciona-se com esta questão ou, de outra forma, o destinatário referido na classificação anterior é, no sistema IIRRA, um processo.

Considere-se, por exemplo, um sistema com uma organização semelhante à da figura 5.4, com um identificador único por processador[1]. Identifiquem-se os tipos diferentes de processos nele existentes (na figura são somente três mas nada impede que as aplicações AP variem de processador para processador). Admita-se ainda que, no mesmo processador, não existe mais do que um processo de cada tipo, mas que cada tipo pode existir em vários processadores (como acontece com os processos DM e TM). Nestas

[1] Valores reservados são utilizados para designar os destinatários de mensagens dos tipos *multicast* e *broadcast*.

circunstâncias, um endereço define-se completamente por dois identificadores, de processador e de processo.

Com a constituição de endereços descrita, é possível dar cumprimento a vários tipos de endereçamento na comunicação de mensagens segundo a classificação de [Hwa93].

Para cada mensagem recebida, o *router* decide, em primeiro lugar, da possibilidade de a mensagem ser endereçada ao processador local. Esta decisão é tomada com base no primeiro nível do endereçamento que identifica o processador ou grupo de processadores destinatários. Se o processador local é endereçado, a mensagem é entregue ao processo cujo tipo corresponde ao identificador no segundo nível do endereço ou a um grupo de processos presentes, se for o caso. Se o processador local não é endereçado ou se o primeiro nível do endereço indica um grupo de destinatários, então a mensagem é reenviada para a rede pelos canais adequados, de acordo com uma tabela de *routing* pré--estabelecida.

Face a estas considerações, uma mensagem constitui-se de duas partes principais, Identificação e Conteúdo (figura5.5).

O conteúdo é obviamente dependente das aplicações e não cabe aqui fazer-lhe qualquer referência.

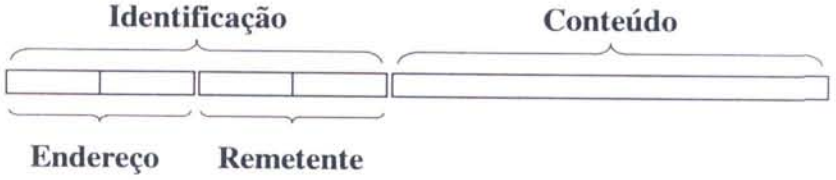


Fig. 5.5 - Constituição de uma mensagem

A identificação da mensagem constitui-se de endereço (especialmente vocacionado para utilização pelos *routers*) e Remetente. O remetente pode ser usado pelo processo receptor da mensagem para identificação, quando necessário, do processo emissor e não tem qualquer função ao nível dos *routers*. Endereço e remetente compõem-se de dois campos cada um, segundo o critério de endereçamento a dois níveis referido.

Com esta constituição de mensagem e de acordo com a classificação de mensagens quanto ao endereçamento [Hwa93], podem definir-se vários endereços especiais, reservando para o efeito alguns identificadores. Nas secções seguintes descrevem-se os endereços GLOBAL e InPathTo implementados, o primeiro do tipo *broadcast* e o segundo do tipo *multicast*.

5.1.1.2 Endereçamento GLOBAL

Dado que se pretende que os *routers* não introduzam qualquer tipo de restrições na topologia da rede de processadores, esta pode ser qualquer e, por conseguinte, o grafo é uma forma possível de a representar. Os nós do grafo representam processadores e os ramos representam canais exteriores ou *links*.

Uma mensagem do tipo *broadcast*, segundo [Hwa93], é enviada por um nó para todos os outros. Corresponde, no sistema IIRRA, ao endereço com identificador GLOBAL.

Numa primeira análise, a distribuição de uma mensagem do tipo *broadcast* concretiza-se efectuando, em cada nó que a receba, a replicação da mensagem por todos os ramos (exceptuando, obviamente, o ramo pelo qual a mensagem foi recebida).

A estratégia anterior é possível se o grafo é acíclico mas não no caso contrário. Através de um ciclo, uma réplica da mensagem acaba por atingir um nó já visitado; efectua-se nova replicação da mensagem e, como resultado final, o número de mensagens geradas é infinito.

O método adoptado para evitar o efeito de proliferação de mensagens baseia-se na definição de *spanning tree* de um grafo [Tar83]. Dado que uma *spanning tree* contém todos os nós do grafo mas não define qualquer ciclo, a replicação de mensagens exclusivamente nos seus ramos garante a visita de todos os nós, gerando um número finito de mensagens. Um exemplo é mostrado na figura 5.6, com a raiz da *spanning tree* colocada no processador central da matriz de processadores.

Assim, uma vez definido o grafo com a topologia da rede e uma *spanning tree* do mesmo, dá-se a conhecer, a cada nó, a sua situação na *spanning tree*, isto é, quais os seus ramos que fazem parte dela. Uma vez em funcionamento, o nó replica, exclusivamente por aqueles ramos, cada mensagem *broadcast* recebida (exceptuando, mais uma vez, o ramo pelo qual a mensagem foi recebida).

Em conclusão, qualquer que seja a topologia do grafo que representa a rede de processaddores, a distribuição de mensagens do tipo *broadcast* é feita seguindo os trajectos definidos por uma *spanning tree* do grafo, de forma a evitar a proliferação de mensagens desnecessárias.

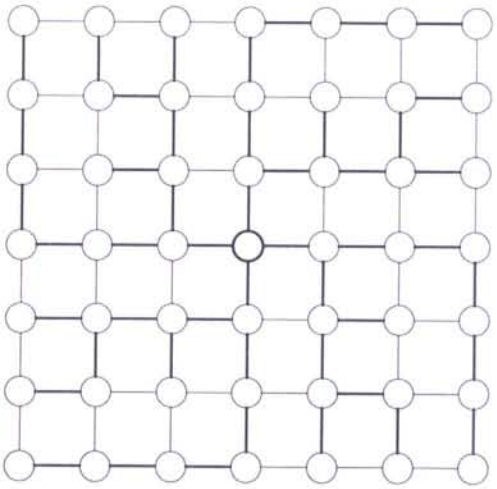


Fig. 5.6 - Uma rede de processadores e a respectiva *spanning tree* (a traço forte) utilizada na distribuição de uma mensagem do tipo GLOBAL

O preço a pagar por tal estratégia é o aumento do tempo de latência. Seja, por exemplo, uma mensagem do tipo *broadcast* enviada por um processador situado num dos extremos laterais do nível mais profundo da *spanning tree*. Antes de atingir o processador situado no outro extremo lateral, a mensagem tem necessariamente que passar pela raiz da *spanning tree*, contabilizando $2.L$ canais (L é a profundidade da *spanning tree*). Possivelmente, existem outros percursos mais directos entre aqueles dois processadores que não podem ser usados na distribuição de mensagens do tipo *broadcast*.

No sistema IIRRA, as mensagens com endereço GLOBAL resumem-se a algumas ordens com origem no processador de coordenação. Devido ao seu pequeno número, o problema mencionado não tem peso significativo. Além disso, dado que a sua origem é fixa, num processador, a distância máxima a percorrer pela mensagem será somente L, desde que esse processador coincida com a raiz da *spanning tree*.

5.1.1.3 Endereçamento *InPathTo*

Um dos pressupostos para o sistema IIRRA é a facilidade de acesso remoto a dados e a tarefas mencionada anteriormente com vista a uma melhor utilização da memória distribuída. Usando esta facilidade, um processador necessitado de um determinado item pode pedi-lo a outro processador ou a um grupo de processadores.

De acordo com uma implementação directa desta estratégia, um pedido, efectuado por um processador P_0 ao processador mais distante P_n, atravessaria d_n canais (*links*), dando origem a outras tantas mensagens (figura 5.7). O tempo de latência seria portanto

proporcional a d_n, o que, para sistemas dotados de um grande número de processadores, consequentemente com um uma rede de diâmetro[1] apreciável, seria demasiado longo.

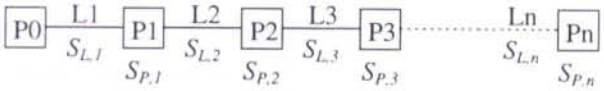


Fig. 5.7 - Pedido de acesso remoto por parte do processador P_0 ao processador P_n

Note-se que o pedido é enviado ao processador P_n porque, de alguma forma, é sabido que ele tem acesso assegurado ao item em falta em P_0, ou seja, a probabilidade S_n de o processador P_n satisfazer o pedido é 1. No entanto e dado que os mesmos dados são acedidos por vários processadores, é provável que algum dos processadores intermédios, no percurrso da mensagem de pedido, possua uma cópia do item em questão. Neste caso, pode obter-se uma resposta antecipada.

Na figura 5.7, seja, $S_{P,i-1}$ a probabilidade de o processador de ordem i-1 satisfazer o pedido effectuado por P_0. Então, a probabilidade $S_{L,i}$ de a mensagem ser relançada pelo processador i-1 através do canal L_i é:

$$S_{L,i} := S_{L,i-1} \cdot \left(1 - S_{P,i-1}\right), \quad com\ S_{L,0} = 1, S_{P,0} = 0 \qquad \text{Eq. 5.1}$$

De onde:

$$S_{L,i} := \prod_{k=0}^{i-1} \left(1 - S_{P,k}\right), \quad com\ S_{P,0} = 0 \qquad \text{Eq. 5.2}$$

Cabe aqui determinar a quantidade de mensagens geradas ao longo do percurso P_0-P_n, por cada mensagem de pedido iniciada em P_0. Seja então um conjunto de M pedidos iniciais. A quantidade de mensagens que, probabilisticamente, atravessam o canal L_i é então:

$$M_{L,i} = M \cdot S_{L,i} \qquad \text{Eq. 5.3}$$

[1] Diâmetro de uma rede de processadores é a maior distância, medida em número de canais, entre qualquer par de processadores.

O total D de mensagens geradas ao longo dos n canais do percurso é:

$$D = \sum_{i=1}^{n} M \cdot S_{L,i} = M \cdot \sum_{i=1}^{n} S_{L,i} \qquad \text{Eq. 5.4}$$

Ou seja, por cada mensagem inicial de pedido de um item, gera-se, na rede, um número médio de mensagens dado por:

$$\overline{D} = \frac{D}{M} = \sum_{i=1}^{n} \prod_{k=0}^{i-1} \left(1 - S_{P,k}\right), \quad com\ S_{P,0} = 0 \qquad \text{Eq. 5.5}$$

O número médio de mensagens geradas no percurso P_0-P_n por cada pedido inicial é equivalente ao percurso médio efectuado pela mensagem de pedido. Por exemplo, se para um pedido efectuado a um processador à distância n=10 resulta um valor $\overline{D} = 3$, então, 3 mensagens são geradas em média e, por conseguinte, a resposta é obtida no processador à distância média de 3 *links* da origem do pedido.

É importante efectuar um estudo da evolução dos valores da distância média $\overline{D}$. Suponha-se que as probabilidades $S_{P,k}$ (excepto $S_{P,0}$) têm todas o mesmo valor S_P. Nestas condições, a equação 5.5 pode ser vista como a soma dos n primeiros termos de uma progressão geométrica com o primeiro elemento unitário e razão 1-S_P, de onde:

$$\overline{D} = \frac{1 - (1 - S_P)^n}{S_P} \qquad \text{Eq. 5.6}$$

Na figura 5.8 apresenta-se um gráfico que representa a evolução da distância média $\overline{D}$ em função da probabilidade S_P de atendimento do pedido num processador intermédio, para vários valores da distância total n.

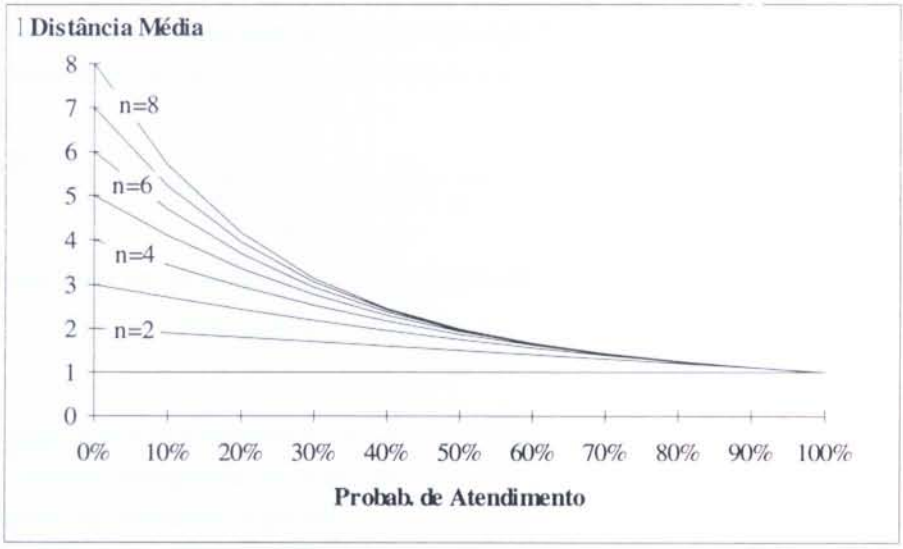


Fig. 5.8 - Distância média percorrida por uma mensagem de pedido de acesso remoto em função da probabilidade de atendimento num processador intermédio

No gráfico da figura 5.9 apresenta-se a evolução da eficiência do método, medida em termos da quantidade percentual de mensagens que são evitadas pelo atendimento antecipado do pedido de um item: $Efic. = 1 - \overline{D}/n$

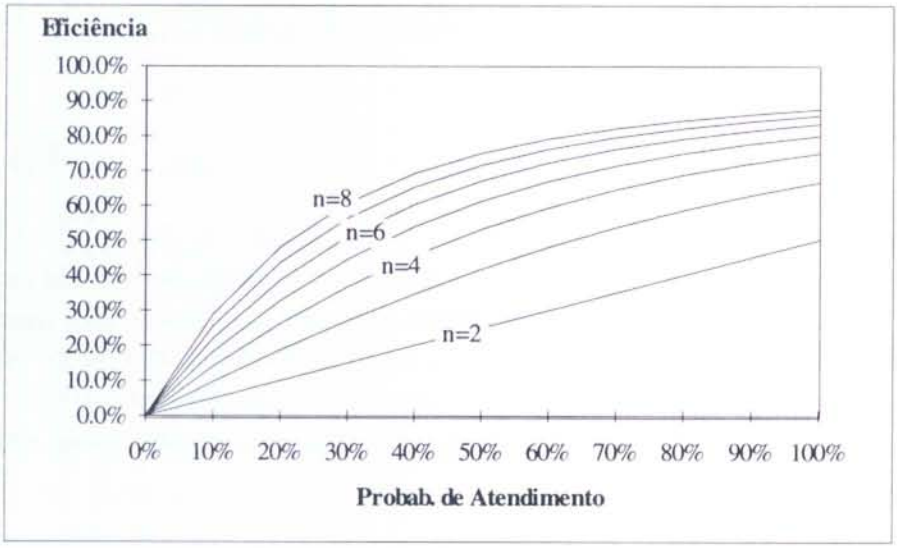


Fig. 5.9 - Eficiência do método de atendimento de pedidos de acesso remoto num processador intermédio

Como se pode ver nas figuras, a redução da distância média $\overline{D}$ e o consequente aumento da eficiência são acentuados nos casos em que a distância total n é grande, mesmo para

valores pequenos da probabilidade S_P. Em redes de pequenas dimensões, consequentemente com baixos valores de n, a melhoria introduzida pela diminuição da distância média não é tão sentida, a menos que a probabilidade S_P aumente para valores próximos da unidade (100%).

Em conclusão, o acesso a dados remotos introduz na rede um número significativo de mensagens, da ordem da distância que separa os processadores inquiridor e inquirido. A provável existência dos mesmos dados em processadores intermédios, encontrados no percurso do pedido de acesso, pode ser utilizada para reduzir a quantidade de mensagens geradas. Tal redução e consequente melhoria na razão Processamento/Comunicações é sentida especialmente em redes complexas, com grandes diâmetros. Em redes de menores dimensões, o mesmo efeito pode ser conseguido, desde que a probabilidade de atendimento do pedido, por parte de um processador intermédio, seja elevada.

A constatação anterior sugere a definição de um tipo especial de endereçamento de grupo (*multicast*), segundo o qual, na sequência de um pedido de acesso a dados remotos, os processadores intermédios possam ser inquiridos acerca dos dados requisitados. O novo tipo de endereçamento designa-se por *InPathTo* [Sou95a].

Para usar as facilidades oferecidas por este tipo de endereçamento, uma aplicação de um qualquer processador, ao efectuar um acesso remoto a um item de um processador particular, coloca no identificador de endereço da mensagem de pedido, uma marca *InPathTo*. A partir desse momento, os acontecimentos são os seguintes:

1. O *router* do próximo processador encontrado na direcção do processador endereçado verifica se uma réplica do processo endereçado (segundo nível de endereçamento) existe localmente e entrega-lhe a mensagem; caso contrário, passa ao ponto 3;

2. É da responsabilidade do processo questionado responder à mensagem, utilizando para o efeito a identificação de remetente ou, na impossibilidade de o fazer, devolvê-la ao *router*, mantendo o mesmo endereço e remetente;

3. Na impossibilidade de obtenção de resposta localmente, o *router* reenvia a mensagem para o processador seguinte, na direcção do processador endereçado na mensagem (regressa ao ponto 1).

Os três pontos referidos vão sendo repetidos até que um processador intermédio possa responder ou, na pior situação possível, até que o processador endereçado seja atingido.

No sistema IIRRA, os principais problemas relacionados com acessos a dados remotos colocam-se ao nível da base de dados de objectos e estes são acedidos de uma forma

bastante aleatória devido à forma distribuída como as amostras são processadas. Realmemte, é possível explorar a coerência dos acessos a objectos localizados junto às fontes dee luz [Hai86], [Hai91] mas outras formas de coerência [Arv88], [Gre89] tornam-se mais difíceis de explorar.

Assim, aa probabilidade de um pedido de objecto por acesso remoto ser atendido num processaador intermédio só tem significado na condição de o objecto se ter mantido em memóriaa local do processador intermédio, durante algum tempo, após ter sido utilizado. A questãão da eficiência da utilização do endereçamento *InPathTo* é assim transportada para os gestores de dados (DM) que serão descritos noutra secção [5.1.2].

5.1.1.4 Deadlock

O probleema de *deadlock* é amplamente conhecido na área dos Sistemas Operativos. Em [Hay88] é definido como sendo "um problema de sincronização segundo o qual um processo se encontra à espera de um evento ... que nunca ocorre". Esse evento pode ser, por exemplo, a acessibilidade a um recurso comum.

Em sisttemas constituídos por processos concorrentes comunicando através de mensageens, o perigo de *deadlock* existe. Realmente, um canal é um recurso comum a um par de prrocessos e pode ser causa de *deadlock*. Neste caso, uma definição mais específica de *deadllock* é dada por [Sil88]: "um conjunto de processos encontra-se em estado de *deadlockk* quando todos os processos do conjunto aguardam um evento que pode ser causado unicamente por um deles".

A comunicação entre dois processos, em OCCAM, efectua-se por meio da transmissão síncrona de mensagens ou seja, não existe memorização de mensagens até que sejam entregues. Assim, o primeiro dos dois processos a ficar pronto para a comunicação bloqueia-se temporariamente até que o segundo processo fique também pronto. Nesse momento, a mensagem é transmitida e os dois processos continuam com a execução da instrução seguinte. Pela definição dada, não há lugar, neste caso, a *deadlock* dado que as instruções de envio e de recepção de mensagens em ambos os processos estão emparelhadas, isto é, cada instrução de envio tem correspondência numa instrução de recepção no outro processo. Se emissão e recepção não se encontram emparelhadas, então um dos processos ficará bloqueado e diz-se que ocorre um *deadlock* algorítmico.

Se dois processos comunicam de uma forma bidireccional, isto é, se enviam mensagens de um para o outro, então pode ocorrer um *deadlock* cíclico, segundo o qual ambos os processos tentam enviar uma mensagem e nenhum deles recebe a mensagem do outro (figura 5.10a). O *deadlock* cíclico pode ocorrer com mais de dois processos.

Efectivamente, qualquer cadeia de envios (ou recepções) de mensagens que defina um ciclo pode criar um *deadlock* que se designa por indirecto (figura 5.10b).

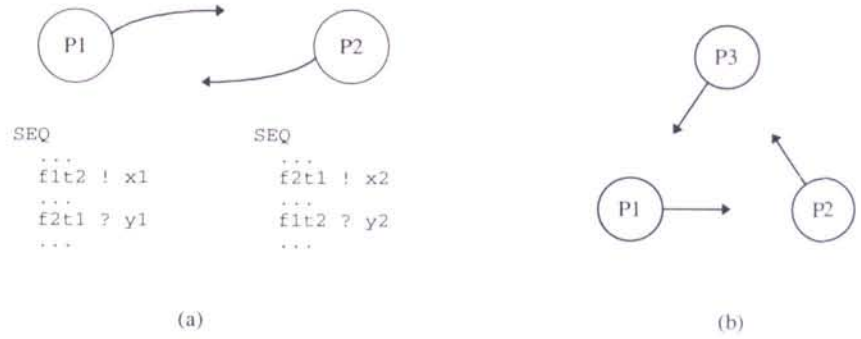


Fig. 5.10 - Exemplos de *deadlock* cíclico

Aparentemente, a memorização das mensagens durante a sua transmissão resolve o problema de *deadlock*. Se, no exemplo da figura 5.10a, existisse um processo de memorização (*buffer*) em qualquer um dos dois percursos, então seria possível enviar uma das mensagens bloqueadas, permitindo ao processo emissor respectivo avançar para outras instruções, nomeadamente a recepção da mensagem que lhe é enviada pelo outro processo. Este ficaria por sua vez desbloqueado, podendo finalmente receber a mensagem que se encontra memorizada.

Realmente, a solução de memorização não resolve o problema, somente o retarda. Repare-se que, no entanto, a memorização é a forma de evitar (não de eliminar) a ocorrência de *deadlock* algorítmico, conforme se constata da análise de uma situação limite: se todos os processos enviam mensagens e nenhum deles recebe, então o *deadlock* acabará sempre por ocorrer. A memorização de mensagens pode no entanto retardar o *deadlock* até ao momento em que (espera-se) um ou mais processos entrem numa fase de recepção.

Em [Sou95a] efectua-se a comparação de dois métodos que evitam a ocorrência de *deadlock* algorítmico ao mesmo tempo que eliminam a hipótese de ocorrência de *deadlock* cíclico. Os dois métodos são utilizados por A. Chalmers [Cha91] e S. Green [Gre91] nos sistemas AMD e DEnIS já mencionados. Ambos os métodos utilizam a memorização de mensagens como forma de evitar o *deadlock* algorítmico, mas diferem substancialmente um do outro.

O método utilizado por A. Chalmers resume-se da seguinte forma:

- dois processos que tentam enviar uma mensagem de um para o outro não entram em *deadlock* se cada um deles, em concorrência com a emissão da mensagem respectiva, efectua a memorização da mensagem com proveniência no outro processo.

O autor refere no entanto que uma implementação à letra do texto anterior não evita *deadlockks* indirectos. Para evitar estes últimos, é necessário que cada processo emissor memorizze todas as mensagens que lhe são enviadas por quaisquer outros processos até ao momentco em que complete a emissão da mensagem. O pseudocódigo correspondente apresenta-se na figura 5.11 e compõe-se de dois subprocessos concorrentes, um de envio e outro cde recepção. A função do canal *to.handle.input* é interromper o subprocesso de recepçãoo no momento em que o processo de envio consegue entregar a sua mensagem. Repare-sse que o processo de envio é executado em alta prioridade de forma a evitar a memorizzação desnecessária de mensagens.

```
PRI PAR

  -- envio de uma mensagem:
  SEQ
    into.neighbour  ! x
    to.handle.input ! any

  -- memorização de todas as mensagens
  -- entretanto recebidas:
  SEQ
    sending := TRUE
    WHILE sending
      PRI ALT
        to.handle.input ? dummy
          sending := FALSE
        ALT k=0 FOR nbr.neighbours
          from.all.neighbours[k] ? y
            ...  put y in buffer
```

FFig. 5.11 - Pseudocódigo OCCAM do método de A. Chalmers para evitar *deadlock*

O métoddo utilizado por S. Green deriva directamente do facto de a rede de processadores por ele utilizada ter uma topologia em árvore ou seja, existe uma hierarquia de processaadores e só existe comunicação de mensagens entre *routers* de processadores hierarquicamente ligados (pais e filhos). Assim:

- um processador pode enviar uma mensagem para o seu ascendente directo se e só se tiver conhecimento que este, ou o canal que lhe dá acesso, se encontra disponível para a transmissão;

- em sentido contrário, não é necessária a verificação de qualquer condição.

O método requer portanto a existência de um protocolo direccional entre cada par de processadores hierarquicamente ligados.

O sistema IIRRA não impõe restrições à topologia da rede e, por conseguinte, a direcção do protocolo deve ser determinada de outra forma, no caso, tendo em atenção a identificação de cada processador:

- um processador com identificação j reporta o seu estado de disponibilidade para a recepção de mensagens a um processador com identificação k se e só se $j < k$.

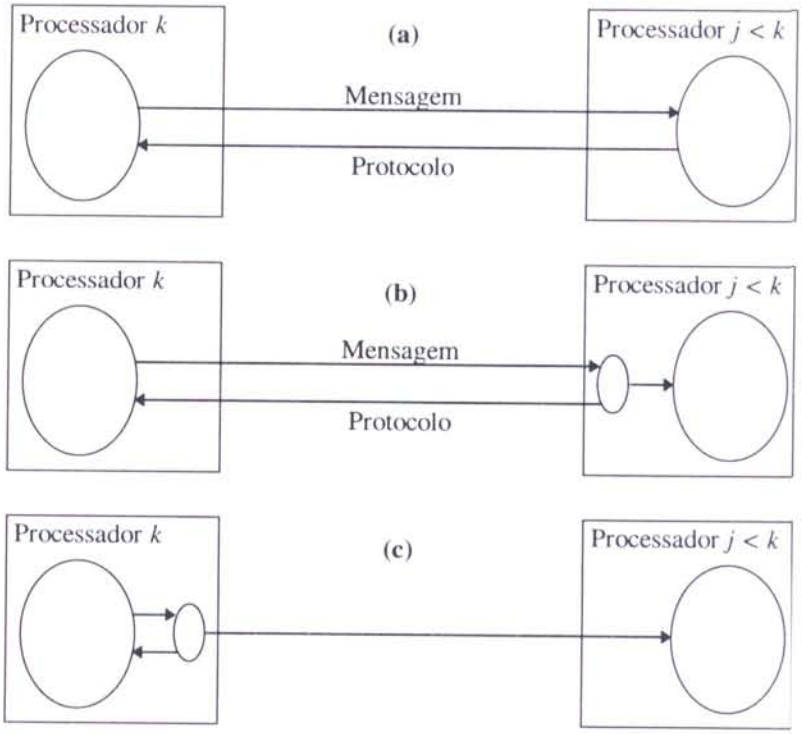


Fig. 5.12 - Redução da quantidade de mensagens nos *links* segundo S. Green

Obviamente, o recurso a mensagens de estado para cumprimento do protocolo duplica o número de mensagens na rede, como se pode ver na figura 5.12(a), diminuindo o seu desempenho. Repare-se que, na mesma figura, o *router* do processador j é o responsável pelo envio do sinal de protocolo para o processador k. Funcionalidade idêntica é obtida pela solução representada na figura 5.12(b), segundo a qual um processo adicional no processador j se encarrega das mensagens de protocolo, libertando o *router* dessa tarefa.

Neste último caso, torna-se indiferente a colocação do processo adicional num ou noutro processador. A sua colocação no processador *k*, como na figura 5.12(c), evita a passagem de mensagens de protocolo através dos canais exteriores, muito mais lentos.

O processo adicional, que se designa por *sender*, recebe mensagens do processo principal do *router* respectivo. Uma vez recebida e memorizada uma mensagem, o processo *sender* fica bloqueado até que consiga enviá-la para o outro processador. Uma vez desbloqueado, envia um sinal de canal pronto para o processo principal do *router*. O pseudocódigo do processo *sender* apresenta-se na figura 5.13.

```
WHILE active
  SEQ
    from.sender.to.main ! any
    from.main.to.sender ? message
    output ! message
```

Fig. 5.13 - Pseudocódico OCCAM de um processo *sender*

O processo principal dos *routers* mantém-se atento à chegada de mensagens do tipo canal pronto com origem nos processos *sender* e de outras mensagens que eventualmente se apresentem pelos canais de entrada. Destas últimas, algumas têm que ser reenviadas, provavelmente por um canal não pronto, pelo que o processo principal mantém listas (FIFO) de mensagens, associadas com os canais de saída geridos por *senders*. Na figura 5.14, apresenta-se o pseudocódigo respectivo[1].

Em resumo, o *deadlock* é evitável, no sistema IIRRA, por qualquer dos dois métodos descritos. Os resultados de experiências efectuadas durante o seu desenvolvimento mostram que, embora algumas variantes de cada um dos métodos possam ser definidas, aparentemente conducentes à obtenção de uma maior eficiência, são realmente os métodos base, tal como foram descritos, os mais eficientes. Dado que essas variantes se baseiam quase sempre na inclusão de mais processos em concorrência, uma justificação para esses resultados pode ser a mesma que foi mencionada na secção 5.1 a propósito da filosofia de programação LSGP ou seja, os *overheads* decorrentes das mudanças de contexto e gestão de filas de processos concorrentes.

[1] Na implementação real, este código é optimizado pela utilização de apontadores. Consegue-se maior rapidez e menor quantidade de memória utilizada no armazenamento de mensagens. Requer um esquema de protecção de memória partilhada não mencionado na figura.

```
WHILE active

  PRI ALT

    -- Atendimento de Protocolos dos Senders
    ALT k=0 FOR n.ways
      from.senders[k] ? dummy
        SEQ
          get.from.fifo(k, message)
          IF
            message exists
              into.senders[k] ! message
            TRUE
              senders.ready[k] := TRUE

    -- Atendimento de Canais de Entrada
    ALT i=0 FOR n.ways
      input IS inputs[i]:
      input ? message
        SEQ
          list.ways.out := determ.ways.out(message.addr)
          SEQ j=0 FOR SIZE(list.ways.out)
            VAL k IS list.ways.out[j]:
            IF
              NOT Sender.Exist(k)
                output IS all.outputs[k]:
                output ! message
              senders.ready[k]
                SEQ
                  to.senders[k] ! message
                  senders.ready[k] := FALSE
              TRUE
                put.in.fifo(k, message)
```

Fig. 5.14 - Processo principal dos *routers* no sistema IIRRA

A decisão de escolha de um entre os dois métodos descritos passa pela sua avaliação em termos de cumprimento dos requisitos ou características mencionados na secção 5.1.1. Considerando que todas as características não mensuráveis são cumpridas por ambos os métodos, resta para avaliação, a taxa de ocupação de processamento por parte dos *routers* e o tempo de latência. Os resultados seguidamente apresentados são extraídos de [Sou95a], onde se efectua a avaliação de ambos os métodos, segundo os dois aspectos mencionados.

A rede de processadores utilizada em [Sou95a] para a avaliação dos dois métodos, é a que se apresenta na figura 5.15. A rede é, do ponto de vista de processadores, homogénea (T805, 25MHz). A traço forte representam-se as ligações entre processadores que definem a *spanning tree* utilizada na distribuição de mensagens com endereço GLOBAL, sendo P0 a raiz da árvore.

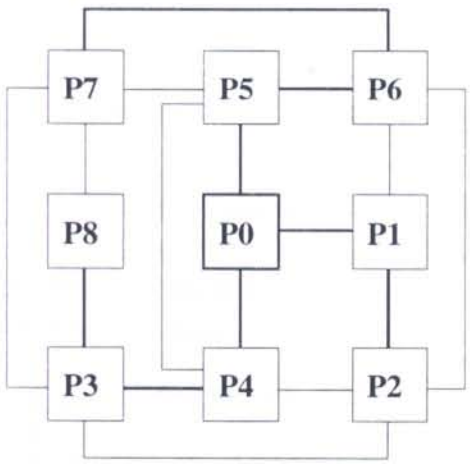


Fig. 5.15 - Rede de processadores utilizada para a avaliação da paralelização do sistema IIRRA

Em cada processador é colocado, em concorrência com o *router* respectivo, um processo cuja função é receber e enviar mensagens para todos os outros processadores. A periodicidade de envio de mensagens é aleatória, assim como os endereços gerados. Desta forma, todos os processadores recebem e enviam aproximadamente o mesmo número de mensagens de e para todos os outros. O comprimento das mensagens é parametrizado de igual forma em todos os processadores de forma a determinar o comportamento dos *routers* com diferentes comprimentos.

Um processo de monitorização determina a taxa de ocupação de processamento da aplicação e do *router* em conjunto. Dado que a aplicação é a mesma em ambos os métodos, eventuais diferenças nas taxas de ocupação são provocadas pelos *routers*.

Devido aos percursos utilizados na transmissão de mensagens, alguns *routers* encontram-se mais sobrecarregados do que outros em termos de mensagens do tipo EE (exterior para exterior, relembre-se). No quadro 5.1 apresenta-se a proporção daquelas mensagens em cada processador. A interpretação a fazer é a seguinte: o processador P8 não recebe mensagens do tipo EE, ou seja, não é incluído, como processador intermédio, em nenhum percurso; o processador P3 consome duas vezes mais tempo com mensagens do tipo EE do que, por exemplo, o processador P1.

No método de S. Green [Gre91] e de acordo com o estabelecimento de hierarquias de processadores para efeitos de protocolo, o número de processos *sender* em cada *router* varia de processador para processador. No quadro 5.1 apresenta-se também o número de *senders* existentes em cada processador, de forma a avaliar a forma como a quantidade de processos concorrentes influencia os tempos consumidos no método de S. Green.

	P0	P1	P2	P3	P4	P5	P6	P7	P8
Proporção de Mensagens EE	2	2	3	4	3	3	3	2	0
Número de Senders	0	1	1	1	3	2	3	3	2

Quadro 5.1 - Proporção de mensagens que atravessam do tipo EE em cada processador e número de *senders* existentes em cada processador

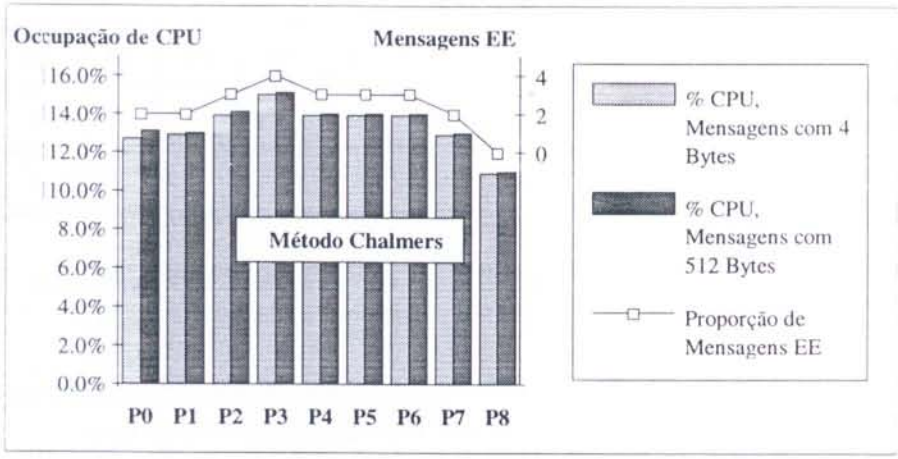
O quadro 5.2 apresenta as taxas de ocupação de processamento nos vários processadores (P0-P8), para os dois métodos e com mensagens de comprimento 4 e 512 *Bytes*.

		P0	P1	P2	P3	P4	P5	P6	P7	P8
Método Chalmers	**% CPU, Mensagens com 4 Bytes**	12.7%	12.9%	13.9%	15.0%	13.9%	13.9%	13.9%	12.9%	10.9%
	% CPU, Mensagens com 512 Bytes	13.1%	13.0%	14.1%	15.1%	14.0%	14.0%	14.0%	13.0%	11.0%
Método Green	**% CPU, Mensagens com 4 Bytes**	12.1%	13.2%	13.7%	14.9%	15.1%	14.5%	15.1%	14.1%	12.2%
	% CPU, Mensagens com 512 Bytes	12.3%	13.7%	14.3%	15.3%	15.7%	14.9%	15.6%	14.5%	12.8%

Quadro 5.2 - Ocupação percentual de CPU dos métodos avaliados, nos vários processadores da rede de teste

A primeira conclusão a retirar do quadro é que as taxas de ocupação de processamento de ambos os métodos não são substancialmente diferentes. Mais ainda, a influência do comprimento das mensagens é pequena, embora seja um pouco superior no método de S. Green.

A figura 5.16 apresenta graficamente resultados dos quadros 5.1 e 5.2 relativos ao método de A. Chalmers. É interessante verificar, na figura, a perfeita correlação de valores entre as taxas de ocupação de processamento no método de Chalmers e a proporção de mensagens do tipo EE a que cada processador dá passagem. Obviamente, a taxa de ocupação de processamento também depende das quantidades de mensagens que o próprio processador envia (mensagens do tipo IE) e que recebe mas, dado que estas são semelhantes para todos os processadores, o seu efeito não é visível no gráfico.



Fiig. 5.16 - Ocupação percentual de CPU do método de A. Chalmers nos vários processadores da rede de teste

No métodoo de S. Green, não se observa a correlação entre as mesmas duas grandezas. Isto signifiica que, neste método, a taxa de ocupação de processamento não é influenciada somente peela quantidade de mensagens do tipo EE. Considerem-se no quadro 5.2, por exemplo, aas taxas de ocupação de processamento, com mensagens de 4 *Bytes*, dos processadoores P0 e P1 (12.1% 13.3% respectivamente). No quadro 5.1 observa-se que, a proporção de mensagens do tipo EE nos mesmos processadores é a igual (2).

Dado que aas aplicações são as mesmas em todos os processadores, as diferenças entre as taxas de prrocessamento dos processadores do exemplo anterior devem ser encontradas ao nível dos *rrouters* ou, mais precisamente, no número de *senders* utilizados. Realmente, no quadro 5.1,, verifica-se que o número de *senders* utilizados nos processadores P0 e P1 é diferente ((0 e 1 respectivamente). Conclui-se portanto que o tempo de processamento associado ao *sender* de P1 (embora mínimo, como se constata pelo pseudocódigo da figura 5.133), assim como os *overheads* associados às mudanças de contexto inerentes a mais um pprocesso em concorrência, afectam globalmente as taxas de ocupação no processadoor[1] e este facto pode ser generalizado a outros processadores.

Em resumoo, no método de A. Chalmers, a taxa de ocupação de processamento de um *router* depeende da quantidade de mensagens do tipo EE que tem que gerir, ou de outra forma, depeende da situação do processador no esquema de *routing* delineado para a rede de processsadores em causa. No método de S. Green, a taxa de ocupação de processameento de um *router* depende também do número de *senders* utilizados, ou seja,

[1] A questão dos *overheads* está de acordo com a opção, referida anteriormente, de desenvolvimento do sistema IIRRA segundo a filosofia LSGP.

da posição do processador na hierarquia estabelecida na rede e que, recorde-se, depende exclusivamente do identificador do processador.

Para avaliação do tempo de latência, o processador P0 (raiz da *spanning tree*) envia uma sequência de mensagens com endereço GLOBAL, e mede-se o tempo que decorre até que todos os outros processadores tenham respondido. No quadro 5.3 apresentam-se os tempos de uma sequência de 10^4 mensagens de comprimentos diferentes e as correspondentes velocidades de transmissão conseguidas com os dois métodos.

	Método Chalmers		Método Green	
Compr. Mens.	Tempo (s)	Velocid. (KBytes/s)	Tempo (s)	Velocid. (KBytes/s)
4	3.6	65.23	4.9	39.55
32	3.7	138.89	5.3	77.98
64	4.2	196.15	5.7	114.23
128	4.5	321.79	6.6	169.68
256	5.5	494.21	8.3	240.79
512	7.5	696.00	11.8	314.04

Quadro 5.3 - Velocidade de comunicação de mensagens do tipo GLOBAL nos dois métodos testados em função do comprimento das mensagens

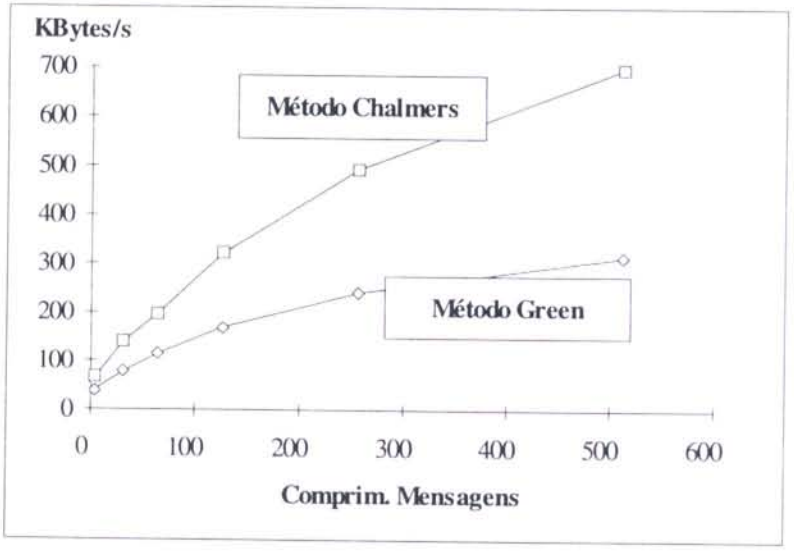


Fig. 5.17 - Velocidade de comunicação de mensagens do tipo GLOBAL, nos dois métodos testados, em função do comprimento das mensagens

A figura 5.17 apresenta, em forma gráfica, a evolução das velocidades de transmissão com o comprimento das mensagens, para os dois métodos. É nítido que o método de

Chalmers tem vantagem, sob este ponto de vista e que a vantagem é tanto maior quanto maior é o comprimento das mensagens.

Em resumo, face aos resultados obtidos por *software* de teste elaborado para o efeito, o método utilizado por A. Chalmers para evitar *deadlock* no sistema AMP é ligeiramente mais eficiente do que o método utilizado por S. Green no sistema DEnIS, quando se trate de mensagens do tipo *unicast*. Ainda para este tipo de mensagens, o método de A. Chalmers apresenta uma menor dependência do comprimento das mensagens e uma dependência moderada da topologia da rede. Além da topologia da rede, o método de S. Green depende também das hierarquias atribuídas aos processadores.

No caso de mensagens do tipo *broadcast*, o método de A. Chalmers é manifestamente mais rápido na sua distribuição.

Tudo parece portanto apontar para a escolha do método de A. Chalmers em detrimento do método de S. Green. O desenvolvimento de aplicações reais sobre módulos de comunicações baseados nos dois métodos levanta no entanto outro tipo de problemas até aqui não mencionados.

Realmente, o método de S. Green revela-se mais estável no que respeita ao enchimento das listas (FIFO) de mensagens por enviar. Neste método, o envio de mensagens por alguns dos canais externos e pelos canais internos é realizado por *senders*, em concorrência com o processo principal, o que torna possível o envio simultâneo de várias mensagens. Por outro lado, enquanto envia uma mensagem por um canal desprovido de *sender*, não recebe outras mensagens. Em consequência destas duas situações, as listas referidas tendem a manter-se com baixos níveis de ocupação.

Pelo contrário, o método de A. Chalmers suporta um só canal de saída de cada vez e, em concorrência, permite a recepção de várias mensagens. O nível de ocupação da lista tende a crescer mais rapidamente do que no método de S. Green, podendo mesmo levar ao abortamento, caso o nível máximo seja ultrapassado.

Em conclusão, o método de A. Chalmers pode ser usado com vantagem em termos de taxas de ocupação de processamento e de tempos de latência, desde que exista um controlo seguro da fluência de mensagens. Assim não sendo, como é o caso do sistema IIRRA, no qual as mensagens de pedidos remotos de dados e de tarefas ocorrem de uma forma imprevisível, o método de S. Green torna-se mais estável, pelo que é o adoptado.

5.1.2 Gestores de Dados

Um dos pressupostos do sistema IIRRA é a sua capacidade de síntese de imagens de cenas complexas, compostas por grande número de objectos. Na versão experimental em arquitectura paralela apresentada na secção [4.2], a descrição da cena é replicada em todos os processadores o que, dada a limitação da capacidade de memória em cada processador, reduz a utilização do sistema a cenas de complexidade média/baixa.

Torna-se necessário definir mecanismos de distribuição de dados pelos vários processadores, de forma a optimizar a utilização da capacidade de memória do sistema como uma entidade global. Os mecanismos definidos devem efectuar o acesso automático a dados localizados remotamente, de uma forma tão transparente quanto possível para as aplicações que deles necessitam. Este problema foi já mencionado na secção 5.1.1.3, a propósito do endereçamento de mensagens *InPathTo*.

Tomando como base a arquitectura da figura 5.4, uma aplicação AP requisita um determinado item ao gestor de dados DM localizado no mesmo processador. Se o processo DM não contém o item requisitado, então, em concordância com os restantes processos DM na rede, obtém-no da forma mais eficiente possível, devolvendo-o à aplicação AP.

A estratégia descrita neste exemplo pressupõe, desde já, uma certa hierarquização dos acessos a memória para obtenção de dados. As diferenças, eventualmente grandes, na velocidade de acesso aos vários níveis da hierarquia exige uma gestão adequada da memória nos níveis de acesso mais rápido. Estas questões, assim como uma descrição do módulo DM de gestão de dados são descritas nas secções seguintes.

5.1.2.1 Hierarquia de Memória

A resolução de problemas de grande dimensão, em sistemas paralelos do tipo multicomputador, exige a distribuição de dados pelas memórias locais dos vários processadores que compõem o sistema. O acesso a dados remotos ou seja, a dados localizados em processadores diferentes, é, por força da lentidão da rede de comunicações, bastante penalizado. Uma vez que os acessos efectuados a dados localizados na memória central dos próprios processadores são substancialmente mais rápidos, interessa criar um mecanismo hierárquico de acessos, segundo o qual são utilizados prioritariamente os acessos mais rápidos e, em caso de insucesso (ausência dos itens endereçados), os mais lentos. Trata-se portanto de um problema que se assemelha ao da gestão de memória virtual em sistemas operativos.

O algoritmo *ray-tracing* apresenta algumas características que favorecem a utilização dos mecanismos mencionados, tal como D. Badouel *et al.* descrevem em [Bad90]:

- "...
- Devido às propriedades de coerência e topologia dos objectos 3D, somente uma pequena parte da base de dados dos objectos é necessária num dado momento. Assim, um mecanismo de *caching* é eficiente para o problema.
- Devido aos efeitos de iluminação (sombras, reflexão e transmissão), a pequena parte da base de dados necessária para o cálculo de uma amostra é bastante difícil de determinar estatisticamente. Torna-se por isso necessário gerir dinamicamente a base de dados.
- O cálculo de uma imagem usa a base de dados somente para efeitos de leitura, pelo que o problema de gestão da coerência dos dados não se coloca."

Partindo destes pressupostos, os mesmos autores desenvolvem um sistema de *ray-tracing* sobre arquitectura paralela multicomputador que descrevem no mesmo trabalho e em [Bad90a], [[Bad94]. O sistema é baseado num esquema de Memória Partilhada Virtual (VSM-*Virtuual Shared Memory*), segundo o qual a memória de dados de cada processador é dividida eem dois blocos, um residente e outro destinado a utilização como *cache*; os blocos residentes de todos os processadores são vistos como um espaço de endereçameento contínuo, organizado em páginas.

No início dde uma sessão de trabalho, a base de dados de objectos, organizada num determinaddo número total de páginas é distribuída pela memória local de todos os processadores. Esta distribuição é feita igualmente, sem qualquer estratégia de optimizaçãoo de acesso. A restante memória em cada processador passa a ser utilizada como memóória *cache*.

Uma vez emm funcionamento, a aplicação de cada processador acede aos objectos através de um móddulo de gestão de memória o qual tenta encontrar um objecto, por ordem de preferência, no bloco de memória residente, na *cache* e, em caso extremo, no processador responsável pelo objecto em causa. Os acessos remotos são efectuados por páginas completas, sendo por isso a memória *cache* gerida em unidades de páginas.

As principaiis observações ao método prendem-se com a distribuição de todos os objectos pela memória total do sistema e com a ausência de uma estratégia para efectuar essa distribuição.

Realmente, o facto de todos os objectos serem distribuídos pela memória local de todos os processadores pode colocar algumas dificuldades na síntese de imagens de cenas com maior complexidade, para as quais seja necessária uma quantidade de memória superior.

Devido à ausência de estratégia na distribuição dos objectos, existe a possibilidade de objectos colocados num bloco de memória residente serem pouco utilizados pelo processador respectivo. Dito de outra forma, o espaço de memória ocupado por esses objectos poderia ser melhor explorado se fosse preenchido adaptativamente. Também o esquema de paginação da memória em páginas de tamanho fixo parece possível de melhoramento dado que, em termos de *cache*, páginas muito utilizadas podem conter itens pouco utilizados.

S. Green [Gre89], [Gre91] utiliza, no sistema DEnIS, um esquema semelhante mas sem objectos residentes em memória. Concentrando-se em questões como a localidade de acessos a dados [Dei84], [Sil88] e exploração da coerência em *ray-tracing* [Arv88], [Gre89], o autor conclui que é suficiente dotar cada processador de facilidades de acesso a itens embebidos no código (itens de frequente acesso e ocupando uma pequena quantidade de memória) e em memória *cache*.

Para o sistema IIRRA, é adequada uma solução do tipo da que é preconizada por S. Green, pese embora o facto de a exploração de coerência em IIRRA ser menos evidente do que em DEnIS. No entanto, pelo menos os objectos na vizinhança das fontes de luz são múltiplas vezes referenciados [Hai86] e [Hai91], o que se traduz em alguma coerência. Acrescente-se ainda que o método utilizado para a aceleração do cálculo de intersecções, baseado na hierarquização de volumes envolventes proposta em [Gol87] e utilizado localmente em outros desenvolvimentos [Cos89], [Fer90], faz aumentar a localidade dos acessos (o método assenta numa estrutura em árvore cujos nós, à excepção das folhas, representam volumes envolventes designados por *clusters* que, por sua vez, são acedidos como se de objectos reais se tratasse).

Para a síntese de imagens de cenas complexas, pressupondo que a descrição da cena não é compatível com a quantidade de memória disponível no sistema, torna-se necessário aceder a memória auxiliar em disco. Este é um recurso único, localizado no computador hospedeiro e com acesso possível por um único processador da rede do sistema[1], o que introduz novos problemas de eficiência [Whi92].

Na figura 5.4, apresenta-se a arquitectura geral do sistema IIRRA e, a propósito, menciona-se que um dos processadores tem funções de coordenação e serviços comuns. Um desses serviços comuns é a interface com o utilizador que, obviamente, pressupõe o acesso a recursos do computador hospedeiro. O acesso do sistema a mais um recurso do computador hospedeiro, o disco, deve portanto ser efectuado pelo processador de

[1] Esta é uma restição imposta pelo sistema de desenvolvimento utilizado.

coordenação e, por este motivo, este é o único processador da rede que tem acesso garantido a todos os dados.

A constatação anterior tem implicações na forma como os processos de gestão de dados acedem a itens localizados remotamente. Em princípio, estes acessos são efectuados utilizando as facilidades de endereçamento *InPathTo* [5.1.1.3]. Desta forma, o processador endereçado deve ser o que possui probabilidade unitária de resposta ao item requisitado. O processador endereçado para efeito de acessos remotos a dados deve portanto ser o de coordenação, dado que somente ele tem acesso garantido a todos os dados, situados em disco.

Como os acessos a disco são significativamente lentos, torna-se necessário acelerar os acessos ao mesmo pela utilização de uma *cache* privada, localizada no processador de coordenação. Esta *cache* deve ser gerida de forma idêntica às restantes *caches*, mas com o sentido de optimizar os acessos remotos efectuados por outros processadores e não pela aplicação local. Os processos de gestão de dados DM localizados na generalidade dos processadores podem assim ser vistos como clientes de um gestor de dados DMS, localizado no processador de coordenação, que age como servidor dos restantes (figura 5.18). A capacidade de resposta deste servidor será tanto maior quanto maior for a quantidade de memória que tiver disponível para efeitos de *cache*.

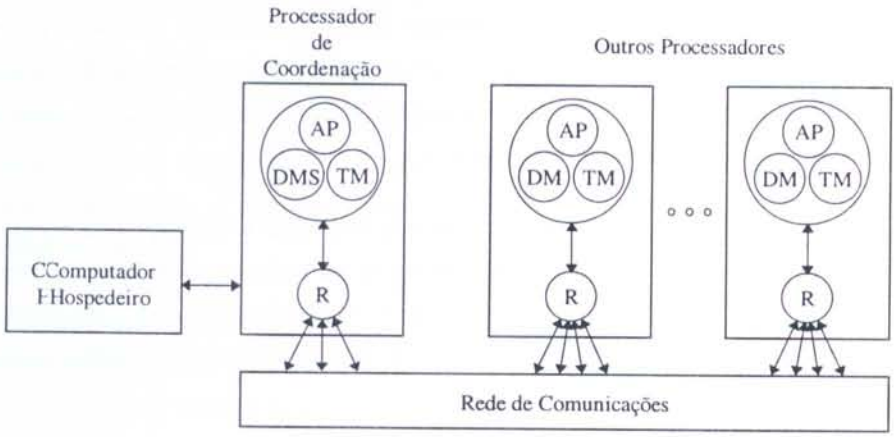


Fig. 5.18 - Atribuição de processos gestores de dados DM e DMS aos processadores no sistema IIRRA

Em conclussão, o sistema IIRRA acede à base de dados de objectos de acordo com um modelo de acessos a dados do tipo Memória Partilhada Virtual, segundo uma hierarquia de quatro níveis:

1. Memória local embebida no código: alguns itens de dados são acedidos com bastante frequência; sendo em pequena quantidade e tamanho, são replicados por todos os processadores da mesma forma que o código de programa;
2. Memória local *cache*: os itens acedidos remotamente podem vir a ser necessários novamente pelo que devem ser memorizados; na impossibilidade de memorização de todos os itens em memória, uma gestão adequada é necessária, explorando a localidade de acessos da aplicação local;
3. Memória remota *cache*: uma vez detectado que um item não se encontra em nenhum dos níveis anteriores, de acesso rápido, um pedido de acesso remoto é emitido para o processador de coordenação, podendo a resposta ser obtida na memória *cache* de qualquer outro processador encontrado no percurso; a gestão da *cache* do processador de coordenação é efectuada de forma a optimizar os acessos que lhe são dirigidos pelos restantes processadores.
4. Memória auxiliar em disco: garante a acessibilidade a um grande número de objectos; só é acedida em último recurso.

5.1.2.2 Gestão de Memória *Cache* Local

Segundo o modelo descrito de acesso a dados do tipo Memória Partilhada Virtual, cada processador, à excepção do de coordenação, é dotado de uma certa quantidade de memória que é gerida como uma *cache* da ou das aplicações locais isto é, em cada momento, são mantidos em memória os dados que estatisticamente têm maior probabilidade de ser referenciados por essas aplicações. Adicionalmente, o conteúdo da *cache* pode ser utilizado para responder a um pedido de um item por parte de outro processador, mas esse pedido não deve alterar a sua gestão.

No caso particular do processador de coordenação, a gestão da *cache* é feita de acordo com os acessos que lhe chegam a partir dos restantes processadores. Desta forma, em prejuízo dos acessos com origem no interior do processador, são mantidos em memória os dados que estatisticamente têm maior probabilidade de ser referenciados pelos outros processadores.

Esta forma de gerir a *cache* do processador de coordenação tem um efeito que é importante realçar: tendencialmente, a *cache* do processador de coordenação é complementar das restantes *caches* e portanto a sobreposição de conteúdos é baixa.

Realmente, a *cache* do processador de coordenação é gerida tendo em conta os pedidos que lhe chegam e esses correspondem, por via do endereçamento *InPathTo*, a faltas de dados em vários processadores. Os dados contidos a dado momento na *cache* do

processador de coordenação tendem a ser dados em falta nos restantes processadores e pode portanto falar-se em complementaridade das *caches*.

Esta complementaridade optimiza o espaço total de memória disponível nas *caches* e minimiza os acessos ao nível hierárquico de memória mais baixo ou seja à memória auxiliar em disco.

Genericamente, a gestão de um *cache* deve garantir um *page fault rate* baixo ou seja, uma pequena percentagem de acessos a itens não contidos na *cache*. O problema principal que se coloca é um problema de substituição e reside na decisão a tomar acerca do item a excluir da *cache* de forma a criar o espaço necessário para incluir um novo item. Obviamente, o item excluído poderá ser requisitado novamente e, nesse caso, gerará uma nova substituição. Algumas estratégias são conhecidas com essa finalidade [Sil88], [DDei84], a maior parte das quais é complexa e de utilidade restrita a casos em que possa ser feita por *hardware*.

Segundo [[Sil88], uma estratégia particularmente adequada a implementações em *software* é a LRU - *Least Recently Used* com implementação baseada numa lista ordenada de acessos[1]. Dada a indisponibilidade de *hardware* especializado que efectue, em cada processador e de uma forma eficiente, uma gestão óptima da *cache* local, os gestores de dados DM em IIRRA utilizam esta implementação.

Ainda segundo [Sil88], a estratégia LRU é uma boa aproximação à estratégia óptima de substituição. Baseia-se no facto de os acessos a dados no futuro próximo serem semelhantes aos acessos efectuados no passado recente. Quando um item necessita de ser excluído, por substituição, a estratégia LRU escolhe o item que não é referenciado há mais tempo.

Na implementação da estratégia LRU em lista ordenada de acessos, sempre que um item é referenciado, o respectivo identificador é movido para o topo da lista. Desta forma, o topo da lista corresponde sempre ao item referenciado mais recentemente e o último elemento da lista ao item referenciado menos recentemente.

O sistema de *ray-tracing* desenvolvido por D. Badouel [Bad90a], [Bad94], utiliza uma estratégia semelhante para gerir o espaço de memória *cache* em cada processador. No entanto, um problema já referido anteriormente diz respeito à forma como os espaços de memória se encontram organizados, em páginas. Não havendo estratégia de integração de vários itens na mesma página (tarefa difícil devido à aleatoriedade com que os objectos são referidos em *ray-tracing*), pode acontecer, por exemplo, que uma página se encontre

[1] Na nomenclatura original [Sil88], é utilizado o termo *stack* em vez de lista ordenada.

no topo da lista devido a um único dos seus itens ser referenciado inúmeras vezes. Por este motivo, os restantes objectos da página ocupam espaço de memória na *cache* mesmo que não possuam uma taxa de referências tão elevada.

Uma estratégia de gestão de memória de *cache* sem recurso à paginação da memória parece ser portanto uma solução mais eficiente. Em IIRRA, os processos DM efectuam essa gestão ao nível do item o que, em termos de *ray-tracing*, pode corresponder a um objecto, a uma fonte de luz, a uma superfície, etc.

5.1.2.3 Concepção do Gestor de Dados

Cada gestor de dados DM ou DMS deve, em consequência do que acaba de ser referido [5.1.2.2], gerir o espaço disponível para *cache* segundo a estratégia LRU. Cada gestor DM efectua essa gestão de forma a optimizar os acessos efectuados pela aplicação local, permitindo simultaneamente acessos remotos, provenientes de gestores DM localizados noutros processadores. O gestor DMS, em situação inversa, optimiza os acessos remotos e permite acessos locais. Salvo referência em contrário, o texto que se segue diz respeito aos gestores de dados DM.

A função principal de um gestor de dados DM é servir a aplicação local, facultando-lhe o acesso, para leitura, a dados considerados de utilização comum, isto é, que podem ser acedidos por qualquer outro processador. A reserva de espaço para dados locais e posterior referência para leitura e escrita é também possível mas com reservas, ficando a cargo do programador a garantia da coerência dos dados decorrente da alteração, em espaços de memória de processadores diferentes, dos mesmos dados.

De acordo com a hierarquia de acessos à memória partilhada virtual, é pressuposto que os acessos de nível 1 (memória embebida no código) são efectuados pela aplicação local, sem intervenção dos gestores de dados. Também o nível 4 (memória auxiliar em disco) é acedido unicamente pelo gestor servidor DMS. Assim, cada processo DM concentra o seu trabalho nos acessos de nível 2 (memória local *cache*) e de nível 3 (memória remota *cache*).

Resumindo, cada DM recebe pedidos da aplicação local para acesso a itens de dados (pedido de nível 2). Se o item se encontra presente na *cache*, a resposta é imediata, caso contrário, uma mensagem de pedido de acesso remoto é enviada para o processo DMS no processador de coordenação (pedido de nível 3), utilizando um protocolo privado e as facilidades de endereçamento *InPathTo* oferecidas pelos *routers*. Neste último caso, os processos DM encontrados no percurso recebem, à vez, a mensagem de pedido (de nível 3) e, caso possuam o item requisitado na respectiva *cache*, devolvem-no, utilizando para

o efeito o remetente da mensagem. Caso não o possuam, reenviam a mensagem para a rede sem alteração dos campos de identificação da mensagem [5.1.1.3]. Também a lista LRU não é alterada, dado que a *cache* é gerida de forma a optimizar os acessos locais (nível 2).

Na fase inicial de síntese de uma imagem, os vários processos DM não contêm quaisquer objectos nas *caches* respectivas. Em consequência, os acessos de nível 3 são em grande quantidade e a rede de comunicações tende a saturar.

Duas situações convergem, no sentido de diminuir a quantidade de mensagens na rede:

- Os processadores que têm uma posição na rede mais próxima do processador de coordenação têm tendência a ser os primeiros a iniciar o processamento e, portanto, a iniciar uma sequência de acessos de nível 3.
- Dada a ausência de objectos nas *caches* dos vários processadores, os acessos de nível 3 incidem preferencialmente sobre o mesmo conjunto de objectos.

É assim frequente que um processador menos afastado do de coordenação desencadeie um acesso de nível 3 a um item e, antes de obter resposta, receba de outro processador mais afastado, um pedido de acesso ao mesmo item. O reenvio deste pedido para a rede é redundante e pode ser evitado pela manutenção, por parte dos gestores de dados DM, de listas de pedidos efectuados por outros processadores.

Tais listas designam-se, no sistema IIRRA, por WRQu - *Waiting for Reply Queue*. Uma lista WRQu é construída sempre que um acesso de nível 2 é convertido em nível 3 e é associada ao item requisitado. Posteriormente, por cada mensagem oriunda da rede com um pedido de acesso àquele item, acrescenta-se o remetente da mensagem na lista WRQu. Uma vez recebida uma mensagem de resposta a um item, efectua-se uma visita à lista WRQu respectiva e replica-se a mensagem recebida para os vários endereços que nela constam.

Esta é uma das técnicas mencionadas em [Cha91] e [Cha93], acerca do sistema AMP, para a minimização da redundância de mensagens relacionadas com acessos a dados remotos. A implementação dessas técnicas é, no sistema AMP, um tanto complexa. Exige uma identificação individual das mensagens e métodos especiais (*snooping* e *poaching*) de detecção de redundância que requerem uma ligação demasiado forte entre *routers* e gestores de dados.

Se bem que as listas WRQu tenham um desempenho especialmente importante na fase inicial do processamento, nada impede que sejam utilizadas com vantagem, durante o restante tempo. Realmente, em termos da equação 5.5, a sua utilização corresponde a

aumentar a probabilidade $S_{P,k}$ de um pedido de acesso a um item ser atendido no processador k.

Em *ray-tracing*, os vários itens a que é necessário aceder organizam-se facilmente em classes (objectos, superfícies, fontes de luz...). É portanto conveniente que a indexação de qualquer item na interface aplicação/gestor de dados seja efectuada de acordo com essa organização.

Organizações mais complexas, segundo uma hierarquia de classes englobando subclasses, são ainda possíveis mas, atendendo a que se trata de uma implementação de gestores de dados em *software*, tornam o sistema demasiado lento. No sistema IIRRA, a hierarquia adoptada possui dois níveis, a Classe e o Item.

A estrutura interna de um processo DM é apresentada na figura 5.19. A estrutura interna do processo servidor DMS é semelhante, à excepção da lista WRQu que não é necessária. Realmente, por razões que se prendem com a não existência de mais processos concorrentes, os acessos a disco são efectuados em modo sequencial com cada pedido ou seja, entre um pedido e a respectiva resposta, não há lugar a atendimento de mais pedidos.

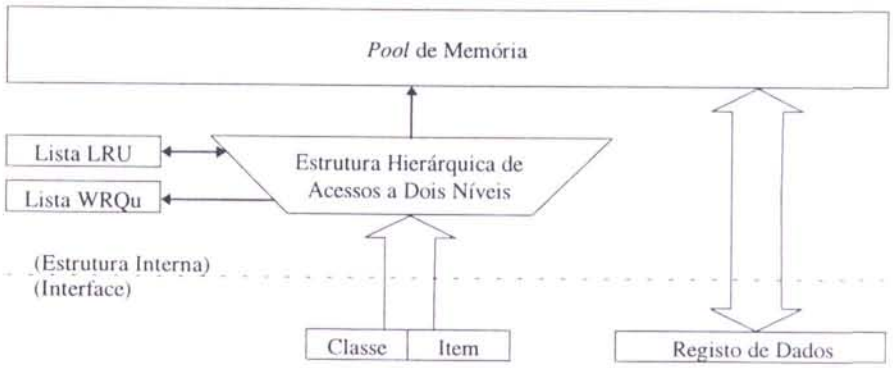


Fig. 5.19 - Estrutura interna de um gestor de dados DM

Uma questão final acerca da concepção dos gestores de dados diz respeito à filosofia de programação LSGP que presidiu ao seu desenvolvimento e ao de todo o sistema IIRRA. Segundo esta filosofia, cada processador contém unicamente dois processos concorrentes, um que gere as comunicações e outro de aplicações que engloba a gestão de dados. Assim, os gestores de dados correm em modo sequencial com as restantes aplicações.

Tal situação cria dificuldades ao facto de cada processo DM (ou DMS) ter que manter uma vigilância permanente à chegada de mensagens provenientes do *router*. A solução

possível baseia-se na visita periódica ao canal respectivo, de forma a determinar da necessidade ou não de lhe dedicar tempo de processamento. A periodicidade de visita é determinada pelos acessos a dados por parte da aplicação, isto é, sempre que a aplicação local efectua um acesso a um item de dados, o processo DM, além de responder à aplicação, verifica se tem mensagens provenientes do *router* para atender.

Esta solução pode ser bastante inconveniente se a aplicação efectua acessos a dados com uma frequência baixa. Tal não é o caso dos processos AP do sistema IIRRA que, como a generalidade dos algoritmos do tipo *ray-tracing*, efectuam inúmeros acessos à base de dados de objectos para efeitos de cálculo de intersecções.

```
PROC dtmc.serve.net(CHAN OF MHPROT from.router, into.router,
                    VAL BOOL waiting.acknowledge,
                    INT w.rec.size, w.rec.ptr)

  SEQ
    serving := TRUE
    WHILE serving
      PRI ALT
        from.router ? mess.size::mess
          CASE type.of.message
            dtm.modify.item.t
              ...  (não descrito)
            dtm.put.item.t
              ...  (não descrito)
            dtm.reply.item.t, dtm.reply.item.undef.t
              -- chegada de um item pedido anteriormente
              SEQ
                IF
                  NOT enought.space.available(pool, <class,item> )
                    ...  exclui último elemento da lista LRU
                  TRUE
                    SKIP
                ...  reenvia mensagem para endereços na lista WRQu
                ...  coloca <class,item> na pool de memória
                ...  calcula w.rec.size, w.rec.ptr para
                --   devolver à aplicação
                serving := FALSE
            dtm.ask.item.t
              -- pedido remoto emitido por outro DM
              SEQ
                st := item.state( <class,item> )
                CASE st
                  exist          -- se existe em memória cache
                    ... responde ao pedido
                  already.asked  -- não existe, já foi pedido
                    ... acrescenta à lista WRQu
                  TRUE
                    ...  reenvia pedido para a rede
        NOT waiting.acknowledge & SKIP
          serving := FALSE
:
```

Fig. 5.20 - Pseudocódigo do procedimento de atendimento da rede num gestor de dados DM

Nas situações limite em que, por falta de tarefas para executar, as aplicações se encontram suspensas, estas devem efectuar periodicamente uma chamada a uma rotina especial no contexto geral dos gestores de dados, de forma a garantir, por parte dos

processos DM e DMS, o atendimento de mensagens com pedidos de acessos remotos a dados. Na base dessa rotina, encontra-se o procedimento de atendimento da rede cujo pseudocódigo se apresenta na figura 5.20 (algumas facilidades do sistema que não se encontram descritas neste texto estão devidamente assinaladas).

O parâmetro `waiting.acknowledge` indica ao procedimento se foi efectuado um pedido de acesso remoto. Em caso afirmativo, a execução do ciclo deve manter-se até à chegada de uma mensagem de resposta (`dtm.reply`). Os parâmetros `w.rec.size` e `w.rec.ptr` devolvem à aplicação local respectivamente o tamanho (em *bytes*) e o endereço do item esperado.

O mesmo procedimento de atendimento da rede é chamado pelo procedimento de interface com a aplicação local para acesso a um item de dados. O pseudocódigo do procedimento de interface apresenta-se na figura 5.21.

```
PROC dtmic.get.data.ptr (CHAN OF MHPROT from.router, into.router
                         VAL INT class, item,
                         INT rec.size, rec.ptr)

  SEQ
    IF
      NOT exist.in.cache( <class,item> )
        VAL waiting.acknowledge IS TRUE:
        SEQ
          ...  acrescenta pedido na lista WRQu
          ...  envia pedido para a rede com endereço InPathTo
          dtmc.serve.net(inp, out, waiting.acknowledge,
                         rec.size, rec.ptr)
      TRUE
        ...  calcula rec.size,rec.ptr para devolver á aplicaçãc
    ...  coloca <class,item> no topo da lista LRU
:
```

Fig. 5.21 - Procedimento de interface do gestor de dados DM com a aplicação local, para acesso de leitura a um item de dados

5.1.2.4 Avaliação

Para se ter uma ideia, ainda que rudimentar, do comportamento do sistema descrito de gestão de memória virtual partilhada para uma aplicação genérica, interessa medir os tempos de acesso da globalidade dos processadores a um conjunto de itens comuns.

A localidade dos acessos a dados é fortemente depende das aplicações propriamente ditas e, por esse motivo, é de difícil simulação. Uma aproximação razoável, principalmente quando as *caches* têm maiores dimensões, pode ser feita com acessos aleatórios. Nesta situação, os vários itens são endereçados por ordem aleatória, diferente em todos os processadores.

Um aspecto importante que influencia o comportamento do sistema é o tamanho dedicado à memória *cache*. É esperado que, *caches* de tamanho inferior produzam piores resultados.

O quadro 5.4 apresenta os tempos de acesso, em segundos, a cerca de um milhar de itens por processador, na rede de processadores representada na figura 5.15, partindo da situação de *caches* vazias. A quantidade de memória *cache* faz-se variar de forma a conter diferentes quantidades de itens e exprime-se, no quadro, na forma percentual da quantidade total de itens. A figura 5.22 mostra os mesmos resultados na forma gráfica.

Percentagem de Itens em Memória Cache							
1%	**2%**	**5%**	**9%**	**18%**	**36%**	**71%**	**100%**
13.70	12.50	11.70	11.80	11.00	8.70	5.20	5.00

Quadro 5.4 - Tempos de acesso aleatório a cerca de um milhar de itens, em segundos, em função da percentagem de itens mantida em *cache*

É de notar que, pelo facto de os itens serem endereçados aleatoriamente, vários deles são acedidos repetidas vezes. Nestes casos, uma *cache* de maiores dimensões favorece a probabilidade de um item requisitado ser encontrado na *cache* local. Mesmo que o item não se encontre na *cache* local, existe alguma probabilidade de ele ser encontrado nos processadores intermédios, antes do pedido atingir o processador de coordenação segundo a hierarquia de acessos de memória implementada [5.1.2.1]. Esta constatação justifica a tendência decrescente dos tempos de acesso, com o crescimento das *caches*.

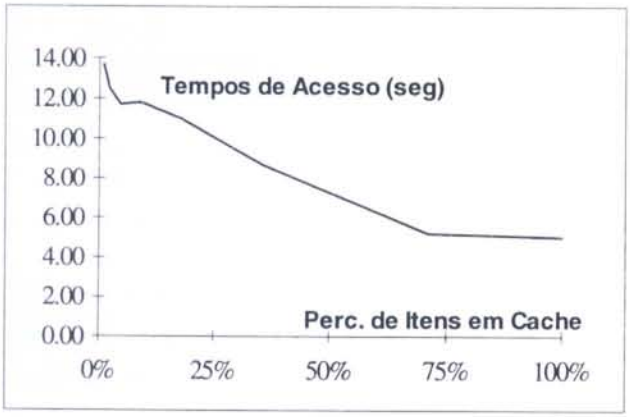


Fig. 5.22 - Tempos de acesso aleatório a cerca de um milhar de itens, em segundos, em função da percentagem de itens mantida em *cache*

Os resultados apresentados dizem respeito a uma situação de início de funcionamento, isto é, correspondem aos tempos de acesso a um determinado conjunto de itens, partindo

da situação de *caches* vazias. É importante avaliar também o comportamento do sistema em regime normal de funcionamento, isto é, partindo da situação de *caches* ocupadas.

Acesso	Percentagem de Itens em Memória Cache		
	9%	71%	100%
1	11.5	5.2	5.0
2	10.8	5.0	2.6
3	11.3	4.9	1.3
4	11.8	4.1	0.7
5	11.7	3.5	0.5
6	11.8	2.8	0.4

Quadro 5.5 - Tempos de acessos sucessivos a cerca de um milhar de itens, em segundos

O quadro 5.5 e a figura 5.23 apresentam os tempos correspondentes a seis acessos consecutivos a cerca de um milhar de itens, endereçados aleatoriamente.

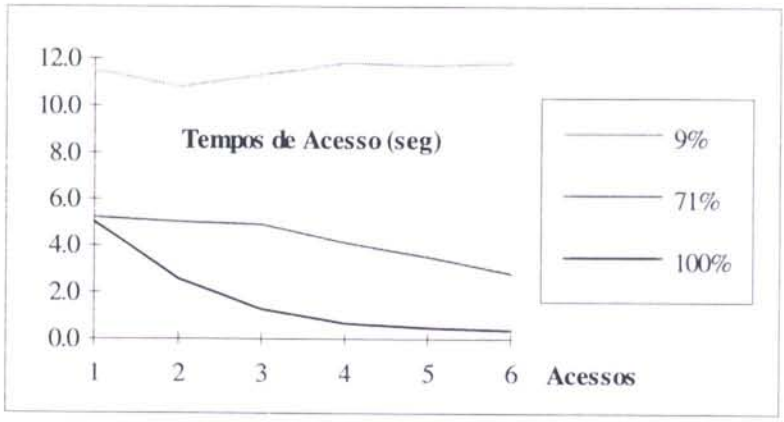


Fig. 5.23 - Tempos de acessos sucessivos a cerca de um milhar de itens, em segundos

Como é esperado, uma *cache* pequena, associada a uma fraca localidade dos dados endereçados, resulta em tempos de acesso sensivelmente constantes. Com *caches* de tamanho suficiente para manter cerca de 71% dos itens, os tempos de acesso decrescem para cerca de metade ao fim de seis acessos.

O gráfico correspondente a uma *cache* de tamanho suficiente para manter todos os itens é também apresentado. Obviamente, esta é uma situação que não é usual, mas permite observar a tendência do comportamento do sistema em situações de forte localidade de acessos. A curva, como pode observar-se, é assintóptica e tende para o tempo de acesso

nas condições de todos os itens se encontrarem disponíveis na *cache* (0.38 seg para os cerca de 1mil itens).

Concluindo, mesmo nas melhores condições de forte localidade de acessos, o tempo médio de acesso por item deverá ser superior a 0.4 milisegundos. Este tempo é demasiado elevado, especialmente se se atender ao facto de a maioria dos itens serem obteníveis a partir de memória local e portanto, sem grandes acessos à rede.

O problema justifica-se pela morosidade da algoritmia utilizada nos gestores de dados, completamente implementada em *software*. Efectivamente, por cada acesso a um item, um processo DM (ou DMS) ocupa uma forte fatia de tempo de processamento na gestão da estrutura de dados necessária à manutenção de informação de utilização dos vários itens. A solução para o problema seria a implementação destes processos em *hardware*, libertando o processador dessas tarefas.

5.1.3 Gestores de Tarefas

Um dos problemas atribuídos à versão em arquitectura paralela descrita na secção [4.2] é a granularidade demasiado fina das tarefas. Como já foi mencionado anteriormente, a solução adoptada no sistema IIRRA redefine uma tarefa como sendo uma célula imagem caracterizada por várias amostras o que aumenta o peso computacional de uma tarefa.

Outro problema, daquela versão é o da gestão centralizada de tarefas e resolve-se, em IIRRA, com gestores locais, designados por TM-*Task Manager*, distribuídos pelos vários processadores de acordo com a figura 5.4. Pretende-se, com esta solução, evitar as comunicações excessivas, mantendo as tarefas nos processadores respectivos, por períodos de tempo superiores.

No início do processamento, a aplicação do processador de coordenação divide o ecrã em células de dimensões semelhantes. As células imagem assim obtidas correspondem às tarefas iniciais e são distribuídas pelos vários processadores do sistema.

De acordo com a noção de realismo crescente, as células imagem são, durante o processamento, subdivididas em outras de menores dimensões. O número de tarefas tende por conseguinte a aumentar durante o processamento. Nas fases finais do processamento, as células imagem vão sendo dadas como totalmente processadas e o seu número diminui até que não existam tarefas para executar. Nesse momento, termina o cálculo da imagem.

No capítulo 4, mostrou-se que, do ponto de vista de *ray-tracing* com realismo crescente, os raios luminosos não possuem todos a mesma importância e, na versão em arquitectura paralela descrita no mesmo capítulo, cada raio corresponde a uma tarefa. Por conseguinte, o processo de coordenação, em conjunto com a base de dados de raios, selecciona a tarefa que, em dado momento, é considerada como mais importante. A selecção corresponde ao primeiro elemento de um conjunto de tarefas ordenadas por ordem de importância.

Na concepção dos gestores de tarefas TM do sistema IIRRA, a ideia de selecção de tarefas por ordem de importância é mantida, mas há que ter em atenção que uma tarefa é agora um conjunto de amostras e não um raio. De acordo com esta afirmação, uma tarefa desencadeia a execução de um conjunto de microtarefas facilmente relacionáveis com amostras e, mais concretamente, com o processamento de raios. De alguma forma, as características destes últimos, nomeadamente a sua importância para a qualidade da imagem, são transferidas para a célula imagem a que dizem respeito e, portanto, um esquema de ordenação de tarefas semelhante ao da versão anterior é executado. As grandezas de controlo de realismo crescente definidas na secção [4.2.2], como sejam as dimensões da célula imagem, a profundidade dos seus raios na árvore de iluminação e o factor de influência desses raios sobre as amostras respectivas são utilizadas na ordenação das tarefas a executar pelos gestores TM.

Ainda na versão em arquitectura paralela descrita na secção [4.2], cada célula imagem é caracterizada apenas por uma amostra. Tal situação impossibilita a exploração da coerência da imagem e dificulta a incidência de maior capacidade de processamento nas áreas da imagem com maior detalhe.

Dado que, no sistema IIRRA, uma célula imagem é caracterizada por um conjunto de amostras, é possível avaliar a coerência das mesmas para, de uma forma semelhante à utilizada na versão sequencial descrita na secção [4.1], subdividir adaptativamente as células imagem e adequar a quantidade de amostras ao nível de detalhe das áreas respectivas. Duas novas grandezas de controlo de realismo crescente são, por conseguinte, utilizadas, o número de amostras da célula e a sua coerência.

Assim, um gestor de tarefas TM, no sistema IIRRA, faz corresponder uma tarefa a uma célula imagem. Mantém uma base de dados de tarefas, ordenadas por ordem decrescente de importância para a qualidade da imagem. A avaliação da importância de uma célula depende das grandezas de controlo de realismo crescente que são:

- a posição (coordenadas) da célula na imagem,
- as dimensões da célula imagem,
- o número de amostras da célula,

- a coerência das células imagem, avaliada por inspecção das amostras que contêm,
- a profundidade dos seus raios na árvore de iluminação,
- o factor de influência desses raios sobre as amostras respectivas.

Os valores de cada uma destas grandezas são devidamente pesados por um conjunto de parâmetros definidos para o efeito.

O número de tarefas geridas por um TM tende a ser, em determinada altura do processamento de uma imagem, bastante elevado, pelo que se colocam questões de quantidade de memória necessária e de eficiência dos algoritmos de ordenação das tarefas nas bases de dados.

Na versão em arquitectura paralela descrita na secção [4.2], estes dois problemas são minimizados pela definição de duas fases de funcionamento, Interactiva e Não Interactiva e pela partição da base de dados, de acordo com as duas fases de funcionamento [4.2.5.1].

Uma solução semelhante é utilizada no sistema IIRRA. Se alguma das grandezas de controlo de realismo crescente que caracterizam uma célula imagem ultrapassa o valor do correspondente parâmetro de controlo definido pelo utilizador, então essa célula não tem qualquer importância para a composição da imagem durante a fase interactiva. As células imagem classificam-se assim em Células em Modo Interactivo (CMI) e Células em Modo Não Interactivo (CMNI) de acordo com os parâmetros de realismo crescente.

Desta forma, cada processo TM gere efectivamente duas bases de dados de tarefas BDI e BDNI (Bases de Dados Interactiva e Não Interactiva, respectivamente). Tal como na versão anterior, somente as tarefas que constam na base de dados BDI necessitam de ordenação, o que torna a algoritmia respectiva mais eficiente. Além disso, a quantidade de memória necessária diminui, dado que as células que constam na base de dados BDNI, não sendo processadas, não são subdivididas e, portanto, não geram descendência.

Em resumo, é da responsabilidade do gestor de tarefas TM manter duas bases de dados de células imagem DBI e BDNI, a primeira das quais é ordenada. Sempre que solicitado para o efeito, o gestor extrai a primeira célula da BDI, e entrega-a à entidade que a solicitou. Também recebe células imagem que classifica em CMI ou CMNI e que deposita na base de dados respectiva.

A descrição anterior formaliza as funções de um gestor de tarefas em termos de necessidades locais, isto é, as funções do processo TM visto como um servidor dos restantes processos do mesmo processador. Realmente, é necessário estabelecer também um conjunto de regras que garantam uma correcta distribuição e, eventualmente, partilha

de tarefas pelos vários processos TM, de forma a obter-se um balanceamento de carga computacional equilibrado.

5.1.3.1 Balanceamento de Carga Computacional

O termo balanceamento de carga computacional de um sistema paralelo é muitas vezes usado como uma grandeza qualitativa que classifica a forma como a carga computacional é distribuída pelos vários processadores. Por exemplo, se os vários processadores de um sistema paralelo começam e acabam simultaneamente a execução de um conjunto de tarefas consecutivas, então todos os processadores executam sensivelmente a mesma quantidade de trabalho e diz-se que o sistema apresenta um bom balanceamento de carga computacional.

O mesmo termo é utilizado para designar a acção ou estratégia de equilibrar o trabalho executado pelos vários processadores do sistema. Neste texto, é este o significado do termo.

Existem basicamente duas formas de balanceamento de carga computacional num sistema paralelo: *à priori* e dinâmico.

Por balanceamento *à priori* de carga computacional entende-se que a distribuição de tarefas pelos vários processadores é efectuada antes de ter início o processamento propriamente dito. Não há lugar a qualquer ajuste na distribuição durante o processamento e daí o nome balanceamento estático pelo qual também é conhecido.

Pelo contrário, um método de balanceamento dinâmico de carga computacional distribui as tarefas pelos vários processadores durante o processamento, utilizando para isso, informação de estado, em termos de carga computacional, de cada processador.

Obviamente que os métodos de balanceamento dinâmico têm tendência a apresentar melhores resultados do que os estáticos mas à custa de um aumento, por vezes importante, nas comunicações inter-processadores. Um trabalho de análise de métodos dinâmicos de balanceamento de carga computacional é apresentado por H. Kuchen e A. Wagener, em [Kuc91].

Os métodos de balanceamento *à priori* são restringidos às aplicações para as quais é possível efectuar uma previsão da carga computacional de cada tarefa. As tarefas são então distribuídas pelos processadores, de forma que o somatório das cargas computacionais previstas das tarefas, seja sensivelmente igual em todos os processadores.

Apresentam-se de seguida dois métodos de balanceamento *à priori* de carga computacional com aplicação possível no sistema IIRRA, Subamostragem da Imagem e Decomposição com Dispersão.

Subamostragem da Imagem

Na secção [2.6.3], descreveu-se o método de balanceamento *à priori* da carga computacional utilizado em [Bad90a] e [Bad94][1] num sistema de *ray-tracing* com subdivisão do volume no espaço objecto [2.6.1].

A primeira parte do método, designada neste documento por Subamostragem da imagem, baseia-se na coerência encontrada entre raios vizinhos. Subdivide o espaço 3D em células paralelepipédicas segundo uma grelha no plano de imagem (ecrã) e emite um conjunto restrito de raios iniciais disperso pela imagem (de onde deriva o nome de subamostragem). Um contador associado a cada célula permite, nesta fase, estimar o tempo que será necessário consumir para o processamento completo de todos os raios[2] que dêem entrada na célula.

Num sistema de *ray-tracing* com subdivisão no espaço imagem, é possível utilizar o método de subamostragem da imagem, com pequenas alterações:

- Divide-se o ecrã em células imagem de iguais dimensões;
- Emite-se um conjunto restrito de raios iniciais (amostras) por célula;
- Processa-se até à exaustão a árvore de iluminação correspondente a cada raio inicial;
- Contabilizam-se, em cada célula imagem, os tempos consumidos pelo processamento anterior.

Dado que existe alguma coerência entre raios iniciais vizinhos e seus descendentes, os tempos contabilizados permitem obter, para as diferentes células imagem, uma razoável estimativa dos tempos médios necessários para o processamento de uma amostra. Sendo fixo o número de amostras por célula, é fácil deduzir o tempo previsto para o processamento completo de uma célula e, a partir dos tempos previstos, é possível agrupar células em processadores, de forma que o somatório de tempos por processador seja sensivelmente o mesmo para todos.

---

[1] Estas referências bibliográficas têm sido efectuadas ao longo deste documento noutros contextos. Realmente, ambas descrevem duas implementações diferentes de *ray-tracing*, uma com sub-divisão no espaço 3D e outra com sub-divisão no espaço imagem.

[2] O termo todos os raios é aqui utilizado, dado que são contabilizados da mesma forma os raios iniciais que entram na célula pelo ecrã e os raios que lhe são transmitidos pelas células adjacentes, nomeadamente, raios iniciais, reflectidos, transmitidos ou sensores de sombra.

O método de subamostragem da imagem não é no entanto adequado ao sistema IIRRA. Realmente, o método baseia-se, por um lado, no processamento exaustivo das árvores de iluminação e, por outro lado, na fixação do número de amostras. No sistema IIRRA, estes dois aspectos são tratados de uma forma progressiva, apresentando portanto algumas dificuldades de compatibilização com o método.

Imagine-se, por exemplo, a seguinte situação limite. O tempo estimado de processamento de uma célula imagem é tão elevado que é a única célula a ser distribuída ao processador respectivo. Em dado momento, os parâmetros de realismo crescente são tais que a célula é classificada em modo não interactivo. Como consequência, o processador fica sem tarefas para execução imediata, restando-lhe ignorar os parâmetros de realismo crescente e continuar a processar a célula (na hipótese de o balanceamento de carga computacional ser exclusivamente estático).

Mesmo salvaguardando o facto de se tratar de uma situação limite, o exemplo descrito mostra que o método de subamostragem da imagem induz a que alguma capacidade de processamento não seja centralizada em tarefas consideradas importantes e portanto, do ponto de vista de realismo crescente, não garante um bom balanceamento de cargas.

Decomposição com Dispersão

Este método baseia-se, tal como o anterior, na coerência que se encontra entre raios iniciais vizinhos ou amostras e no facto de a coerência ser tanto maior quanto menor for a distância entre as amostras.

A afirmação anterior implica que, numa área rectangular com P amostras, a coerência máxima obtenível para o conjunto de amostras se encontra, tendencialmente, quando o total das distâncias medidas entre amostras é mínima e, por conseguinte, quanto a área coincide com um quadrado de dimensões $\sqrt{P} \cdot \sqrt{P}$ amostras.

No sistema IIRRA, a unidade de tarefa não é a amostra mas sim a célula imagem. No entanto, substituindo os termos amostra e área rectangular, por célula imagem e macrocélula, respectivamente, pode deduzir-se uma extrapolação e, com ela, definir uma regra de balanceamento da carga computacional:

- Numa macrocélula composta por P células imagem de dimensões iguais, a coerência máxima obtenível para o conjunto de células encontra-se, tendencialmente, quando a macrocélula coincide com um quadrado de dimensões $\sqrt{P} \cdot \sqrt{P}$ células imagem;
- Fazendo P igual ao número de processadores e atribuindo uma célula imagem a cada processador, obtém-se, em média, uma boa aproximação à melhor distribuição das tarefas da macrocélula;

- Repetindo o processo para as restantes macrocélulas da imagem, obtém-se, em média, uma boa aproximação ao equilíbrio óptimo da carga computacional dos vários processadores, para toda a imagem.

A figura 5.24 resume, graficamente, a regra anterior, para um sistema constituído por 16 processadores.

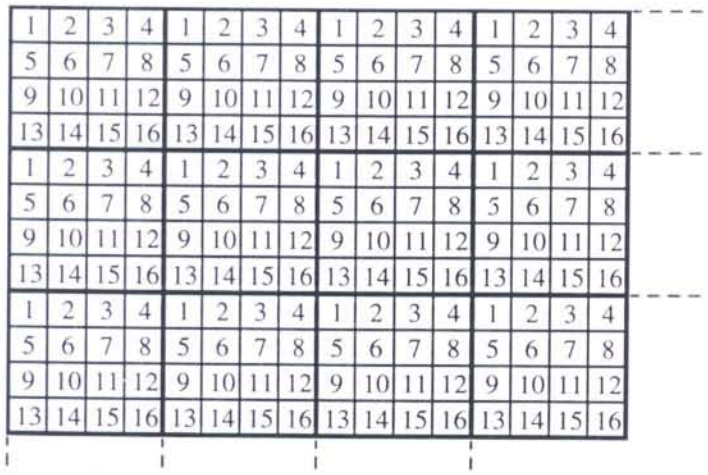


Fig. 5.24 - Balanceamento da carga computacional pelo método da decomposição com dispersão

Um método semelhante é referido em [Fol90], sob a designação de *Interleaved Partitioning*, para efectuar o balanceamento *à priori* da carga computacional em sistemas paralelos de rasterização de imagem.

Em [Kob89], é utilizado um sistema semelhante mas em 3D, sob a designação de alocação distribuída [2.6.2]. Em [Gre91], o método toma a designação de Decomposição com Dispersão (*Scattered Decomposition*) e é analisado do ponto de vista de aplicação em *ray-tracing*. Pelos resultados apresentados, o sistema apresenta diferenças razoavelmente pequenas na carga computacional de cada processador quando, para os exemplos que apresenta, o número de macrocélulas geradas (chamadas regiões, na referência) é igual ou superior a 32x32.

Em termos de aplicação no sistemas IIRRA, o método de decomposição com dispersão não apresenta os mesmos problemas de incompatibilidade com a noção de realismo crescente que o método de subamostragem apresenta. Realmente, se uma célula imagem é muito complexa, as células vizinhas tendem também a sê-lo, pelo que, em termos médios, a distribuição de complexidade é feita equitativamente por todos os processadores.

Existe no entanto um outro problema de compatibilização, este relacionado com as dimensões das células imagem. Para efeitos de balanceamento de carga computacional, as células imagem devem ser pequenas. Para efeitos de realismo crescente, admite-se que as células são subdivididas adaptativamente e, portanto, as células iniciais devem ter dimensões mais generosas.

Há que estabelecer um compromisso nas dimensões das células de forma a não prejudicar nenhum dos dois aspectos. De qualquer forma e até porque o método de decomposição com dispersão já é, por si, um método aproximado, alguns desequilíbrios na carga computacional acabam sempre por surgir.

Conclui-se então que um método de balanceamento de carga *à priori* não garante uma distribuição equitativa de trabalho pelos vários processadores e que, sendo assim, um método dinâmico é necessário.

H. Kuchen e A. Wagener, em [Kuc91], classificam os métodos de balanceamento de carga computacional dinâmicos em:

- Métodos Passivos: os processadores requisitam tarefas a outros mais sobrecarregados;
- Métodos Activos: os processadores distribuem as tarefas que criam;
- Métodos Mistos: combinação dos dois anteriores.

As vantagens dos métodos passivos, ainda segundo os mesmos autores, são:

- Enquanto todos os processadores têm tarefas suficientes, os *overheads* relacionados com o balanceamento de cargas são mínimos;
- O processamento inerente ao balanceamento de cargas é executado, principalmente, pelos processadores com pouca ou nenhuma actividade.

A principal desvantagem, é que, nas fases inicial e final do processamento, a quantidade de tarefas é globalmente pequena e os pedidos de tarefas atrasam os processadores activos.

S. Green, em [Gre91], apresenta dois algoritmos de gestão dinâmica de tarefas que se enquadram nos métodos passivos de balanceamento de carga computacional. A diferença entre os dois algoritmos é basicamente a granularidade das tarefas. Enquanto que um define uma tarefa como sendo indivisível e equivalente a uma área de ecrã, o outro define-a como sendo divisível em subtarefas correspondentes a raios.

Ambos os métodos se baseiam na topologia da rede do sistema paralelo utilizado, em árvore. Assim, o conjunto global de tarefas, em número superior ao de processadores, é

determinado *à priori* e partilhado por todos os processadores, na base de um processador pedir tarefas ao seu precedente na árvore, sempre que seja necessário. Cada pedido passa de processador em processador até atingir a raiz da árvore que, em resposta, envia uma nova tarefa. Cada nó da árvore mantém um contador de pedidos por cada descendente directo (leia-se subárvore) de forma a poder distribuir as tarefas pelos processadores inactivos. Os tempos de latência são minimizados pela memorização, em cada processador, de algumas tarefas para distribuição.

A estratégia utilizada por S. Green é muito semelhante à que é utilizada pelos sistemas em *farm* de processadores [May89] como é o caso da versão em arquitectura paralela [Sou95] mencionada no capítulo anterior [4.2.8.1].

D. Badouel *et al.*, em [Bad90a] e [Bad94], utilizam também um método passivo de balanceamento dinâmico da carga computacional, mas algo diferente do método de S. Green. No início do processamento, o ecrã é dividido em partes iguais, tantas quantos os processadores do sistema. Com uma granularidade de tarefas tão grossa, obviamente que haverá sempre um processador que termina a execução da sua tarefa bastante mais cedo do que os restantes.

Há então lugar a uma sequência de mensagens, segundo a qual, um processador inactivo envia um pedido de tarefas aos restantes processadores. Um destes responde com uma pequena parte da tarefa que tem em processamento (3x3 *pixels*, segundo os autores).

Para efeitos de partilha de tarefas, os processadores encontram-se dispostos em anel. Assim, um pedido de tarefas percorre sequencialmente os processadores, até que seja satisfeito, ou até que o processador que o emitiu seja atingido. Neste último caso, não existem tarefas em execução e portanto a imagem está calculada.

Em conclusão, embora este aspecto de balanceamento dinâmico da carga computacional não se encontre estudado de uma forma exaustiva no sistema IIRRA, parece que nenhum dos métodos descritos, se aplicado directamente, serve os seus propósitos.

O método de S. Green é demasiado dependente da arquitectura, em árvore. Em consequência, torna difícil a partilha de novas tarefas criadas pela subdivisão das células imagem, nos processadores mais longínquos do de coordenação.

O método de D. Badouel *et al.*, mais uma vez, parte do princípio que uma tarefa (ou subárea do ecrã), uma vez iniciada, é processada exaustivamente, o que não acontece no sistema IIRRA.

Um método de gestão dinâmica de tarefas baseado na partilha entre processadores vizinhos, do tipo dos mencionados em [Kuc91] parece ser adequado. Facilidades de *routing*, existentes ou a definir, podem ser neste caso uma ferramenta preciosa.

De qualquer forma, os métodos dinâmicos tendem a aumentar o tráfico de mensagens na rede que, no sistema IIRRA, se encontra já bastante carregada com mensagens relacionadas com a base de dados em memória partilhada virtual. Por conseguinte, há que efectuar alguns esforços no sentido de diminuir a actividade da gestão dinâmica de tarefas.

Neste contexto, um método de balanceamento *à priori* da carga computacional como o de decomposição com dispersão, distribui tarefas iniciais de complexidade semelhante pelos vários processadores do sistema. Dado que a criação de novas tarefas corresponde à subdivisão das anteriores no espaço imagem, a complexidade das novas tarefas tende a ser igualmente distribuída e portanto, as diferenças de carga computacional tendem a ser pequenas.

A gestão dinâmica das tarefas limita-se assim a corrigir aquelas diferenças, pelo que a sua actividade não deverá ser muito elevada.

5.2 Melhoria do Crescimento do Realismo

O algoritmo *ray-tracing* é, reconhecidamente, bastante consumidor de tempo de cálculo. A forma sequencial como efectua o cálculo de uma imagem, *pixel* a *pixel*, agrava a situação pois o utilizador fica obrigado a aguardar o processamento completo, para avaliar uma imagem que, eventualmente, rejeitará.

Uma via de resolução para o problema, baseada na obtenção de imagens intermédias, com níveis de qualidade progressivamente melhores, foi definida no capítulo 4. Genericamente, corresponde a fazer progredir a qualidade da imagem no tempo e, por isso, designa-se por *Ray-tracing* com Realismo Crescente.

A versão sobre uma arquitectura paralela apresentada na secção [4.2] foi desenvolvida como meio experimental de avaliação das noções introduzidas e que, resumidamente, se traduzem na evolução da imagem, tanto em resolução como em efeitos ópticos, estes últimos por avanços progressivos na profundidade das árvores de iluminação. O utilizador pode assim iniciar o cálculo de uma imagem com uma resolução grosseira e com árvores de iluminação limitadas a uma profundidade baixa, obtendo, como resultado, uma aproximação rápida à imagem final.

Com maiss tempo de processamento, o utilizador pode refinar a resolução e/ou permitir o processamento de mais raios reflectidos e transmitidos por forma a incluir mais efeitos ópticos. Para o efeito, dispõe de um conjunto de parâmetros que lhe permite controlar a resolução,, a quantidade a qualidade dos efeitos ópticos e a definição de áreas de interesse.

Pela avaliiação efectuada, conclui-se que os principais problemas a resolver, em termos de realismo crescente, são relacionados com a resolução. A imagem é dividida em áreas rectangulaares ou células imagem e cada uma destas é caracterizada por uma única amostra. Não existindo, na estrutura de dados, qualquer relação entre as várias amostras, a cor deterrminada para uma amostra é replicada por todos os *pixels* da célula respectiva.

Desta estrratégia resulta que as primeiras aproximações da imagem, com uma resolução mais grosseira, apresentam um acentuado efeito de mosaico. Além disso, a falta de ligação emtre as várias amostras impede a verificação de eventuais sinais de coerência entre elas,, sem a qual não é possível transferir o esforço computacional de zonas da imagem ccom pouco detalhe para outras de maior detalhe.

5.2.1 Reedução do Efeito de Mosaico

Na soluçãão sequencial de *ray-tracing* com realismo crescente de J. Leitão *et al.*, apresentadla na secção [4.1] [Lei91], [Lei93], o problema do efeito de mosaico é minimizaddo por meio de uma interpolação bilinear dos pontos de fronteira de cada célula imagem.

Por cada ceélula imagem é calculada apenas uma amostra, situada num dos seus cantos e a estrutura dde dados implementada permite aceder às amostras das células vizinhas. A quantidadee de amostras disponíveis por cada célula é, por conseguinte, igual ou superior a quatro (nno exemplo da figura 5.25, a célula (a) tem disponíveis oito amostras).

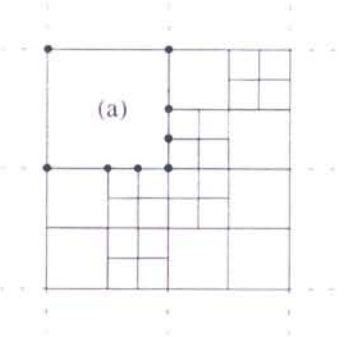


Fig. 5.25 - Subdivisão adaptativa das células imagem segundo J. Leitão *et al.*

Uma interpolação bilinear explorando a informação de todas as amostras disponíveis é executada e daí resulta uma continuidade de grau C^0 nas linhas de fronteira das células. O efeito de mosaico é eliminado, passando eventualmente a observar-se, nas fronteiras das células, o efeito de *Mach band* [Fol90], [Wat92], [Gla95].

Estas bandas, muito mais subtis, são indiscutivelmente melhor aceites pela vista humana do que as descontinuidades de grau C^{-1} correspondentes ao efeito de mosaico inicial. Além disso, devido ao refinamento da resolução em realismo crescente, a sua existência é apenas temporária, pelo que não se justifica a adopção de métodos de interpolação mais precisos [3.1].

A interpolação bilinear descrita apresenta no entanto um efeito de alastramento de cor semelhante ao que se obtém em desenho tradicional quando se arrasta o carvão pela pressão de um dedo. Tal efeito deve-se ao facto de todas as amostras disponíveis para a interpolação se encontrarem na fronteira da célula imagem, não existindo amostras interiores à mesma.

T. Akimoto *et al.*, em [Aki91], apresentam uma variante de *ray-tracing* que designam por PSRT (*Pixel Selected Ray-tracing*), com a qual pretendem acelerar os cálculos de intersecções, explorando a coerência encontrada em grandes áreas da imagem.

O método começa por efectuar uma amostragem grosseira da imagem, segundo uma malha rectangular, o que pode ser comparado ao cálculo de uma amostra por célula imagem, em *ray-tracing* com realismo crescente. Por classificação e comparação de quatro amostras, nos cantos de uma região rectangular, o método decide se o ponto médio, ao centro desta, necessita de ser calculado por *ray-tracing* ou simplesmente interpolado[1]. Quando necessário, as regiões rectangulares são subdivididas em outras de

[1] Omitem-se alguns detalhes técnicos relacionados com a classificação e comparação das amostras, por se considerar que não se situam no contexto do assunto em análise.

menores dimensões pelo que, pontos anteriormente interpolados, passam a ser efectivamente calculados.

O método de T. Akimoto *et al.* tende a minimizar o efeito de alastramento de cor do método de J. Leitão *et al.* dado que, quando necessário, inclui uma amostra no interior da área rectangular interpolada. No entanto, não faz um aproveitamento exaustivo das amostras disponíveis na vizinhança dessa área. Para a obtenção de melhores resultados, deve assim definir-se um método de interpolação que, de alguma forma, reúna as vantagens destes dois.

No sistema IIRRA, uma célula imagem é caracterizada por um número significativo de amostras, algumas das quais se situam na fronteira da célula e, outras, no seu interior. A distribuição das amostras pela área da célula deve, dentro do possível, definir um padrão regular, de forma a evitar as estruturas de dados complexas que possam desfavorecer o factor paralelização.

Também por motivos de redução da complexidade da estrutura de dados de uma célula imagem, admite-se, na presente implementação do sistema IIRRA, a não inclusão de transparências.

Um dos pressupostos da noção de realismo crescente é a distribuição equitativa do esforço computacional necessário para o cálculo de amostras, em áreas com uma coerência semelhante,. Dito de outra forma, a localização das amostras na imagem, deve ser feita de forma a evitar a concentração excessiva de amostras em determinadas áreas, em detrimento de outras, que possuam o mesmo grau de coerência.

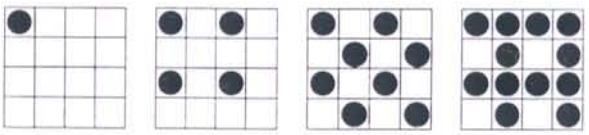
Não faz portanto sentido que uma célula imagem, uma vez definida, veja todas as suas amostras processadas de uma só vez. É necessário estabelecer uma ordem pela qual as amostras são processadas e, eventualmente, agrupá-las de forma a estabelecer níveis de detalhe dentro de uma célula imagem.

Se as várias amostras são afixadas directamente na imagem, sem qualquer tipo de interpolação entre elas (equivalente a células imagem uniponto [4.2.9]), a imagem compõe-se dos pontos cujas amostras são conhecidas. O problema de estabelecer uma ordem de processamento de amostras tem, neste caso, a ver com a capacidade de integração espacial da vista humana e é portanto comparável ao problema de preenchimento de matrizes de pontos em técnicas de *dithering*.

Segundo [Fol90], existem várias possibilidades para as matrizes de *dither* e as matrizes de desenvolvidas por B. Bayer [Bay73] minimizam o efeito de textura introduzido nas imagens pelas técnicas de *dithering*. Segue-se um exemplo de matriz de *dither* de ordem quatro.

$$D^{(4)} = \begin{bmatrix} 1 & 9 & 3 & 11 \\ 13 & 5 & 15 & 7 \\ 4 & 12 & 2 & 10 \\ 16 & 8 & 14 & 6 \end{bmatrix} \qquad \text{Eq. 5.7}$$

Segundo este exemplo, o nível N de cinzento obtém-se preenchendo, numa matriz quadrada de pontos, aqueles que, na matriz 5.7, possuem numeração de ordem inferior ou igual a N. Os casos (a), (b), (c) e (d) da figura 5.26 representam os preenchimentos da matriz correspondentes aos níveis de cinzento 1, 4, 8 e 12, respectivamente.



(a) (b) (c) (d)

Fig. 5.26 - Matrizes de *dither* para quatro valores diferentes de níveis de cinzento

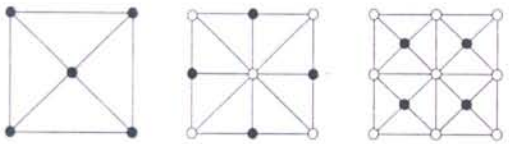
Se, no sistema IIRRA, as amostras de uma célula imagem são calculadas pela ordem definida na matriz 5.7, pode utilizar-se uma estratégia de cálculo de amostras repartido pelas células. Uma solução possível seria a definida pelos seguintes passos:

1. Seleccionar uma célula imagem carenciada de amostras,
2. Calcular um número restrito de amostras (uma só, no caso limite),
3. Devolver a célula à base de dados respectiva e passar ao ponto 1.

Esta estratégia garante uma distribuição uniforme das amostras por toda a imagem ou pelo menos pelas áreas dotadas de uma coerência semelhante. Em contrapartida, fica dificultada a tarefa de interpolar as amostras no interior de uma célula imagem.

Uma alternativa a esta estratégia, orientada para a interpolação da imagem nos espaços entre amostras, baseia-se na triangulação das células imagem, fazendo coincidir as áreas a interpolar com triângulos e os pontos de amostragem com os vértices dos mesmos. As amostras são agrupáveis e cada grupo de amostras define um nível de amostragem diferente. De cada vez que uma célula é seleccionada, um grupo completo de amostras é processado.

Um exemplo de processamento de uma célula, com três níveis de amostragem, é apresentado na figura 5.27. Os pontos a cheio significam as amostras correspondentes ao nível de amostragem em questão.



(a) (b) (c)

Fig. 5.27 - Exemplo de processamento de uma célula com três níveis de amostragem

A partir da figura, podem verificar-se dois aspectos que, pela sua importância do ponto de vista de realismo crescente, interessa realçar: cada nível de amostragem constitui-se sensivelmente do mesmo número de amostras; as áreas dos triângulos em cada nível de amostragem são iguais.

A igualdade de área dos triângulos produz (principalmente nas fases iniciais de cálculo, em que as células têm grandes dimensões) uma imagem regular, com igual espaçamento das áreas interpoladas. Reduz-se assim o aspecto de textura criado por células imagem vizinhas, contendo, cada uma delas, áreas triangulares diferentes.

O facto de os níveis de amostragem possuírem aproximadamente o mesmo número de amostras, permite uma abordagem consistente da resolução, em termos de realismo crescente. Seja a situação de todas as células classificadas em modo interactivo se encontrarem em condições idênticas, do ponto de vista das grandezas de controlo de realismo crescente. Uma vez seleccionada uma célula imagem, são processadas todas as suas amostras do nível de amostragem adequado e o mesmo procedimento é repetido para as restantes células (seguindo a solução anterior, a três passos).

A resolução aumenta de igual forma em toda a imagem, num espaço de tempo que é proporcional ao número de amostras do nível de amostragem processado. O passo seguinte, supostamente um novo aumento de resolução, processa aproximadamente o mesmo número de amostras por célula e consome aproximadamente o mesmo tempo.

Na figura 5.28 apresenta-se uma sequência de imagens obtida com o sistema IIRRA. As imagens possuem uma resolução de 128x128 *pixels*. Como se pode observar, o efeito de mosaico é substancialmente reduzido, se comparados com as imagens obtidas com a versão experimental sobre uma arquitectura paralela [4.2.9].

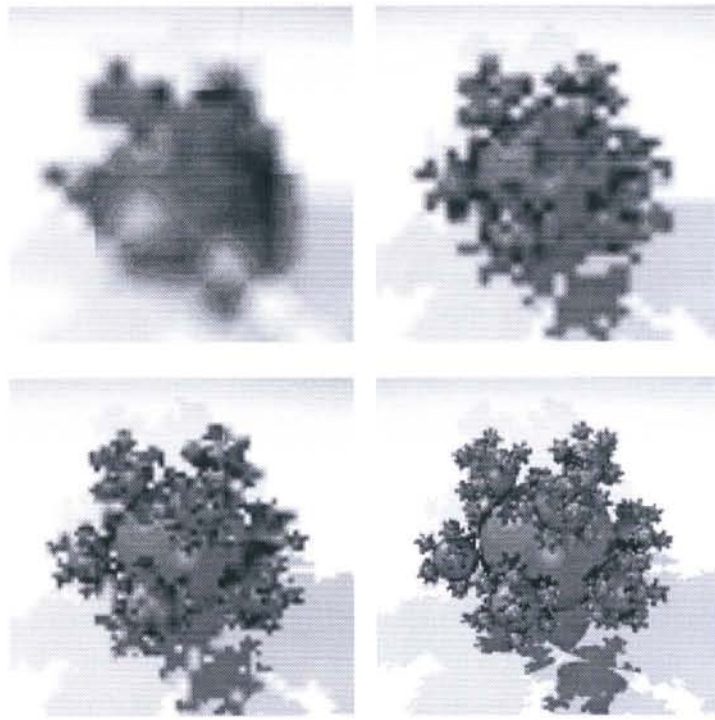


Fig. 5.28 (Original na Secção a Cores) - Sequência de imagens por progressão em resolução no sistema IIRRA

Em conclusão, o esquema de amostragem da figura 5.27 garante que, para um determinado conjunto de células em igualdade de circunstâncias em termos de grandezas de controlo de realismo crescente, o aumento de resolução é efectuado numa sequência de três níveis cuja duração individual é sensivelmente a mesma. Comparativamente com a solução em arquitectura paralela da secção [4.2], as células imagem podem tomar dimensões maiores e, dado que o interior das células é interpolado, o efeito de mosaico tende a ser bastante menor.

O esquema de amostragem facilita ainda a subdivisão adaptativa das células imagem. Repare-se, na figura 5.27(c), que as quatro pequenas áreas quadrangulares possuem uma caracterização idêntica à área inicial de 5.27(a). É portanto possível subdividir uma célula imagem em quatro menores, sendo cada uma destas processada, posteriormente, da mesma forma. Amostras já calculadas são automaticamente passadas de célula mãe para célula filha, de forma a minimizar a repetição de cálculos efectuados.

O problema da repetição dos cálculos não se encontra ainda totalmente resolvido. A paralelização do sistema dificulta, por questões de eficiência das comunicações, a utilização de estruturas complexas de dados. Por este motivo, uma célula imagem, uma vez subdividida, transforma-se em quatro novas células totalmente independentes, o que dificulta a partilha de resultados de amostras idênticas, mas localizadas em células

diferentes. É necessário efectuar alguma investigação no sentido de avaliar as vantagens e desvantagens da utilização de estruturas poderosas de interligação de células imagem (uma hipótese óbvia é a *quadtree*), que permitam, por um lado, partilhar amostras na fronteira de células vizinhas e, por outro lado, efectuar a troca de células entre processadores, de forma a conseguir-se um razoável balanceamento da carga computacional.

5.2.2 Exploração da Coerência da Imagem

A coerência encontrada em áreas extensas da imagem pode ser um precioso auxiliar para o crescimento do realismo. Áreas coerentes podem ser amostradas a uma taxa inferior a outras dotadas de maior detalhe de imagem.

A subdivisão adaptativa das células imagem referida na secção anterior [5.2.1] é um meio possível de explorar a coerência referida. Carece no entanto de um meio de decisão que permita ao sistema determinar quais as células imagem a subdividir e qual o melhor momento para o fazer.

J. Maillot *et al.*, em [Mai92], apresentam um método estocástico de amostragem da imagem que permite a progressão da resolução no tempo. Numa área da imagem, equivalente a uma célula imagem do sistema IIRRA, é lançado um número crescente de raios iniciais, cujas árvores de iluminação são processadas exaustivamente.

O número de amostras é crescente e, por cada nova amostra processada, o método recorre a um teste SPRT (*Sequential Probability Ratio Test*) [Wal48] para decidir se a célula imagem respectiva é homogénea, não homogénea ou se o número de amostras não é suficiente para a tomada de decisões.

Uma célula imagem classificada como homogénea é afixada no ecrã, enquanto que uma não homogénea é subdividida em quatro menores, repetindo-se o processo para cada uma delas. Se o número de amostras não é suficiente, continua o processo de criação de raios iniciais no interior da célula imagem em análise.

Se bem que bastante poderoso, o método de amostragem descrito apresenta alguns problemas. Um, de carácter genérico, é que o método falha na classificação de células homogéneas, em alguns casos em que o contraste no interior das mesmas é muito elevado. Como consequência, permanecem alguns erros na imagem, detectáveis, principalmente, nas linhas de contorno dos objectos.

Outro problema, mais relacionado com a sua aplicabilidade no sistema IIRRA, diz respeito ao número de amostras necessárias por célula imagem que, em células de maiores dimensões, é muito elevado. As mensagens necessárias para a partilha de tarefas (células imagem) entre processadores seriam demasiado longas, com efeitos negativos sobre as comunicações.

Também a aleatoriedade do posicionamento das amostras no interior de uma célula imagem dificulta a triangulação desta para efeitos de interpolação.

O método utilizado em [Lei91], [Lei93], para a decisão de subdividir ou não uma célula imagem, baseado na sensibilidade da vista humana ao contraste [Mit87] parece ser adequado (equação 4.7). A implementação é directa, pois é suficiente a actualização dos valores máximo e mínimo por cada componente de cor, de cada vez que se evolui no processamento de um grupo de amostras de uma dada célula. A definição de um limiar de contraste a partir do qual as células são subdivididas passa a corresponder a um novo parâmetro de controlo de realismo crescente: homogeneidade das células imagem.

Na figura 5.29 apresenta-se uma imagem, obtida num estágio bastante precoce da sequência de realismo crescente, onde se pode observar a subdivisão adaptativa das células imagem.

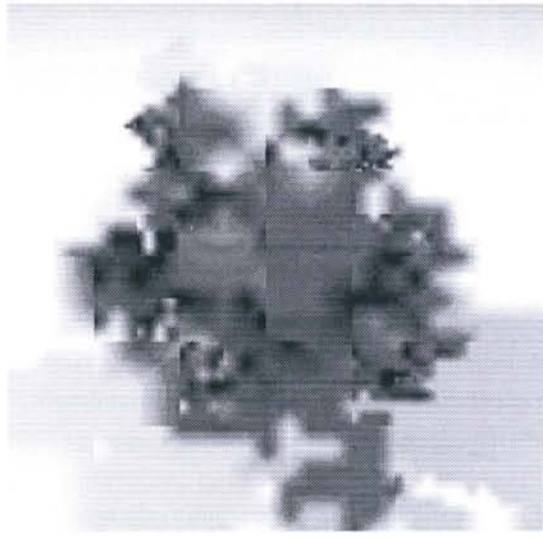


Fig. 5.29 (Original na Secção a Cores) - Subdivisão adaptativa de células atendendo ao seu contraste

Obviamente que, em células de grandes dimensões, com o reduzido número de amostras utilizado, o valor de contraste calculado possui um intervalo de confiança demasiado largo. Nestes casos, pode acontecer que uma célula mais homogénea seja subdividida

antes de uma célula menos homogénea, o que não é grave. A situação contrária, de uma célula ser classificada erradamente como homogénea, pode levar a resultados intermédios de pior qualidade, dado que, ao não ser subdividida, impede a observação imediata de alguns detalhes. Eventualmente, as facilidades concedidas ao utilizador na definição de áreas de interesse podem amenizar o problema, embora de uma forma não automática.

5.3 Síntese das Soluções Encontradas

No capítulo 4, apresentou-se uma versão experimental de *ray-tracing* sobre uma arquitectura paralela dotado de capacidade de síntese de imagens com realismo crescente. Ao longo do presente capítulo, enumeraram-se os principais problemas encontrados com a versão experimental e definiu-se um novo sistema, IIRRA, com a finalidade de ultrapassar aqueles problemas. Os vários módulos constituintes do sistema foram estudados e vários métodos para a sua implementação foram analisados e avaliados, com referência especial às soluções adoptadas. A presente secção pretende efectuar uma síntese dessas soluções, de forma a permitir uma visão global do sistema IIRRA.

Genericamente, a arquitectura do sistema é representada na figura 5.4. Compõe-se de um conjunto de processadores, interligados por uma rede de comunicações, qualquer que seja a sua topologia. Os processadores comunicam entre si através de mensagens, de forma a partilharem informação de dados e de tarefas. Por conseguinte, em cada processador existem, além da aplicação propriamente dita, mais dois processos, um de gestão de dados e outro de gestão de tarefas, que libertam a aplicação local dos problemas relacionados com essa partilha. Um processo adicional, designado por *router*, coordena as comunicações de cada processador com os restantes.

A aplicação local AP de cada processador efectua os cálculos relacionados com o algoritmo *ray-tracing* propriamente dito, tendo em atenção as noções de realismo crescente definidas no capítulo 4. Assim, a imagem é inicialmente subdividida em células imagem, as quais são distribuídas pelos vários processadores.

Cada célula imagem é caracterizada por um conjunto de amostras, no máximo de treze e distribuídas por níveis de amostragem, segundo a figura 5.27. Cada amostra corresponde a um raio inicial e consequente árvore de iluminação, cujo processamento é efectuado também por níveis de profundidade na árvore. Um esquema de subdivisão adaptativa das células permite explorar a coerência da imagem, com base no cálculo do contraste, de forma a incidir o esforço computacional nas zonas da imagem dotadas de maior detalhe.

Do ponto de vista de realismo crescente, cada célula imagem caracteriza-se por um conjunto de grandezas, ditas de controlo do realismo crescente, algumas das quais são importadas da versão experimental da secção [4.2]:

- posição da célula na imagem (coordenadas de posição, necessárias à definição de áreas de interesse),
- dimensões da célula (em número de *pixels*),
- número de amostras em processamento (ou nível de amostragem),
- factor de homogeneidade da célula (contraste encontrado nas amostras),
- profundidade, na árvore de iluminação, dos raios correspondentes às amostras em processamento,
- máximo factor de influência detectado nos raios correspondentes às amostras em processamento.

De acordo com os valores destas grandezas, o processo de gestão de tarefas TM [5.1.3] define a ordem pela qual as células são processadas, de forma a obter o melhor efeito possível, em termos de realismo crescente. Tal como na versão experimental, cada grandeza de controlo do realismo crescente é associada com um parâmetro definido pelo utilizador que funciona como valor limite da grandeza respectiva. Uma célula é classificada em modo não interactivo desde que, pelo menos uma das grandezas referidas ultrapasse, em valor, o do respectivo parâmetro. Uma célula classificada em modo não interactivo é depositada, pelo gestor de tarefas, numa base de dados não ordenada (BDNI). As restantes células são consideradas importantes para o realismo da imagem e são mantidas em outra base de dados (BDI), por ordem de importância.

Assim, a aplicação local processa uma célula imagem, por avanço no nível de amostragem ou por avanço na profundidade da árvore de iluminação, consoante a situação da célula em causa. Entrega-a ao gestor de tarefas que a reclassifica e insere na base de dados adequada e que lhe devolve a próxima célula a processar, em termos de realismo crescente.

É também da responsabilidade do gestor de tarefas TM efectuar o balanceamento da carga computacional [5.1.3.1]. Um processo TM, na ausência de tarefas consideradas importantes, deve requisitar aos processos do mesmo tipo, dos processadores vizinhos, uma ou mais tarefas para processamento local. Realmente, a gestão dinâmica das tarefas encontra-se ainda em fase de investigação, pelo que o balanceamento referido se limita, de momento, a um balanceamento *à priori* do tipo Decomposição com Dispersão [Gre91].

Durante o processamento de uma célula imagem, a aplicação local AP necessita de aceder aos dados de descrição da cena 3D como sejam descritores de objectos, de

superfícies e de fontes de luz. Dado que a quantidade de informação envolvida pode ser muito elevada, o sistema IIRRA faz uso de um esquema de Memória Partilhada Virtual, segundo uma hierarquia de memória a quatro níveis [5.1.2.1].

Um processo gestor de dados DM é replicado em todos os processadores e é da sua responsabilidade disponibilizar à aplicação local os itens de dados por ela requisitados [5.1.2]. Para o efeito, mantém numa *cache* local os itens mais utilizados, segundo uma estratégia LRU [5.1.2.2]. O espaço de memória da *cache* é gerido ao nível do item, de forma a evitar os inconvenientes da paginação.

Sempre que um determinado item requisitado pela aplicação local não exista em memória *cache*, o processo DM emite uma mensagem de pedido de acesso a dados remotos, dirigida a um gestor de dados servidor DMS situado no processador de coordenação (figura 5.118). Este gere uma memória *cache* de tamanho bastante superior às restantes pelo que, com grande probabilidade, responde rapidamente ao pedido. Além disso, o processo DMS mantém em disco uma imagem de todos os itens de dados que corresponde ao último nível da hierarquia de memória partilhada virtual.

Os acessos a dados remotos são obviamente lentos pelo que uma optimização se torna necessária. Um pedido de um item é transmitido de processador em processador até atingir o de coordenação. Cada processador intermédio, no percurso, é questionado e, se possível, responde ao pedido. Este procedimento é automatizado pela utilização das facilidades de endereçamento *InPathTo* previstas nos *routers* [5.1.1.3]. Além disso, vários processadores podem tentar aceder simultaneamente ao mesmo item. Neste caso, um esquema de redução de pedidos redundantes é previsto em cada processo DM [5.1.2.3].

De acordo com as descrições anteriores, a maior parte das mensagens são dirigidas para o processador de coordenação. Em termos de comunicações, e apesar das optimizações conseguidas, esta estratégia tende a sobrecarregar os processadores mais próximos daquele.

O processador de visualização, sendo um recurso único não paralelizável, é também bastante solicitado em termos de comunicações. Realmente, o processo de visualização tem que receber, das restantes aplicações, a informação necessária para a actualização da imagem.

Assim, existem dois grandes fluxos de mensagens no sistema IIRRA, um em direcção ao processador de coordenação e outro em direcção ao processador de visualização. Obviamente, a rede de comunicações será tanto mais eficiente quanto menores forem as sobreposições encontradas nos percursos dos dois fluxos.

A forma mais simples de minimizar essas sobreposições é colocar na rede, os dois processadores em causa, em posições, que sejam diametralmente opostas. A ideia é fazer coincidir os canais e processadores mais sobrecarregados por um dos fluxos de mensagens com os menos sobrecarregados pelo outro fluxo.

A arquitectura do sistema, genericamente proposta na figura 5.4 e melhor detalhada na figura 5.18, fica finalmente definida segundo a figura 5.30.

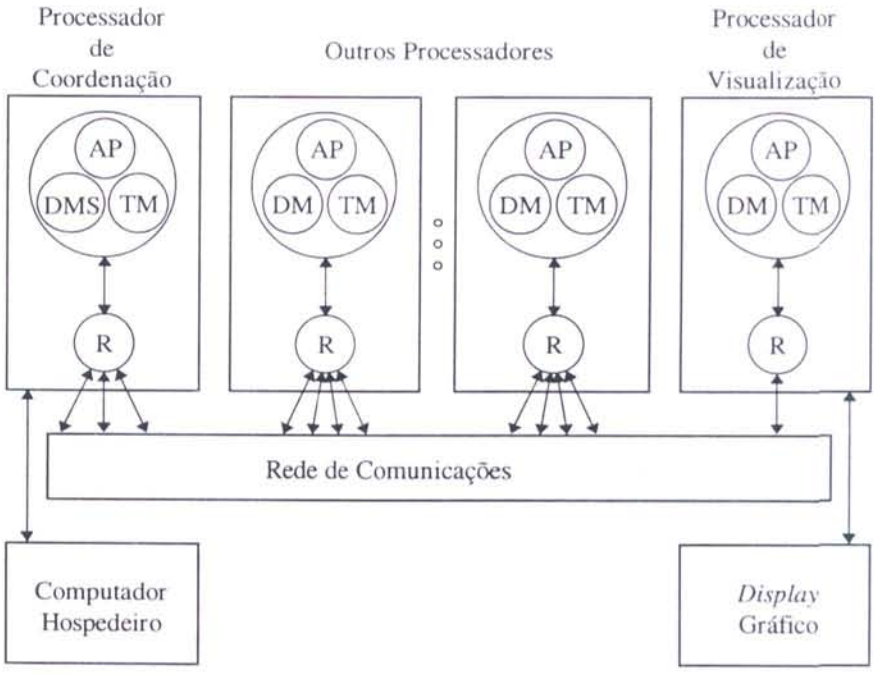


Fig. 5.30 - Arquitectura do sistema IIRRA comportando o *display* gráfico

O facto de o processador de visualização ser um recurso único não paralelizável tem ainda outras consequências, relacionadas com a complexidade das tarefas que tem de executar e com a cadência máxima a que consegue executá-las.

As tarefas relacionadas com a visualização devem ser simples e de rápida execução de forma a que o refrescamento da imagem possa acompanhar os restantes processadores. Assim, todos os procedimentos relacionados com a afixação da imagem devem ser executados de uma forma distribuída, até ao momento em que um *pixel* é depositado no *frame buffer*. A solução adoptada baseia-se na manutenção, em cada processador, de *frame buffers* virtuais, um por cada célula imagem em processamento local. O processador de visualização limita-se assim a receber, dos restantes processadores, blocos inteiros de imagem que copia para o *frame buffer* real.

No entanto, à medida que o número de processadores no sistema aumenta, a cadência segundo a qual os *frame buffers* virtuais são actualizados, aumenta (espera-se) na mesma proporção, podendo chegar ao ponto em que o processador de visualização satura todo o sistema. Com um ligeiro incremento nas comunicações, um sistema de *polling* evita a saturação do sistema, fazendo coincidir a cadência de envio de *frame buffers* virtuais com a cadência permitida pelo processador de visualização. Assim, o processador de visualização envia mensagens de pedido, sequencialmente, para os restantes processadores, sempre que esteja disponível.

Um aspecto de grande importância no sistema IIRRA prende-se com a comunicação de mensagens entre processadores.

Um *router* é um processo replicado em todos os processadores, que serve os restantes processos locais com facilidades de recepção e envio de mensagens de e para outros processos localizados em outros processadores [5.1.1]. Os processos clientes não conhecem necessariamente a topologia da rede, pelo que é da responsabilidade dos *routers*, o encaminhamento das mensagens, de processador em processador, desde a sua origem até ao seu destino, seja para mensagens *unicast*, *multicast* ou *broadcast*.

A comunicação entre processos através de mensagens é susceptível de *deadlock*. Dois métodos que evitam o *deadlock* são apresentados e devidamente avaliados [5.1.1.4]. Em termos de eficiência, a maior diferença encontrada entre os dois métodos localiza-se ao nível da transmissão de mensagens do tipo *broadcast* que não possuem um peso importante no contexto do sistema IIRRA. Assim, o método implementado nos *routers* do sistema IIRRA para evitar o *deadlock* é o proposto por S. Green [Gre91] por se ter apresentado mais estável, quando em condições reais de funcionamento.

Capítulo 6

Conclusões

6. Conclusões

O presente documento foi elaborado no âmbito de uma Dissertação de Doutoramento na área da Computação Gráfica. Pretendeu-se com ele apresentar os trabalhos mais importantes desenvolvidos nesse contexto, dedicados, genericamente, à Síntese de Imagens Realistas por Computador.

Neste capítulo efectua-se um resumo das principais conclusões acerca desses trabalhos, e perspectivam-se desenvolvimentos futuros sobre temas afins.

6.1 Conclusões Genéricas

A Síntese de Imagens é uma área de investigação em franco desenvolvimento. Procura-se, hoje em dia, obter a perfeição na simulação dos fenómenos que regem a interacção da energia luminosa entre vários objectos, possibilitando, tanto quanto possível, a obtenção de imagens tão realistas quanto uma fotografia. Os algoritmos de Iluminação Global contabilizam esses fenómenos, de uma forma realista, mas com um custo computacional demasiado elevado para uma utilização em ambientes que se pretendam interactivos. Os assuntos abordados no primeiro capítulo focam exactamente este aspecto, utilizando, para medida, um parâmetro qualitativo que se designou por Relação Custo Computacional/Realismo da Imagem ou simplesmente relação custo/realismo dos métodos de síntese de imagem.

Nesse contexto, referiram-se os algoritmos de visibilidade com cálculo de iluminação local como possuindo um custo/realismo baixo ou médio. Alguns destes algoritmos são bastante utilizados de uma forma integrada em sistemas de CAD (*Computer Aided Design*) para permitir a visualização dos objectos modelados.

Outras aplicações, algumas interligadas com os sistemas de CAD, são mais exigentes em termos de realismo da imagem. Começa a ser comum, na indústria dos dias de hoje, sintetizar imagens de produtos novos como forma de reduzir o fabrico das amostras, morosas e caras. Esta possibilidade permite partir de um leque mais largo de ideias e chegar a um conjunto mais restrito de produtos a analisar. Mais ainda, a síntese de imagens a partir de modelos concebidos localmente em sistemas de CAD pode ser utilizado em acções de *marketing* de novos produtos, desde que o nível de realismo esteja de acordo com o fim em vista.

Para estas aplicações, os algoritmos de iluminação global são necessários, mas a relação custo/realismo que lhes está associada é demasiado elevada. Justificam-se, desta forma, esforços no sentido de diminuir o custo computacional desses algoritmos, sem perda significativa do realismo das imagens que produzem, ou, em alternativa, manter o custo computacional mas aumentando a capacidade de processamento, de forma que os tempos de cálculo diminuam.

A segunda alternativa baseia-se, obviamente, na utilização de vários processadores, para resolverem, em conjunto, o mesmo problema ou seja, no recurso a sistemas de arquitecturas paralelas de computadores.

Em conclusão, duas vias são exploradas nos desenvolvimentos referidos neste documento. Uma, é a introdução de melhorias à algoritmia, independentemente da arquitectura utilizada. A outra é a via da paralelização.

No segundo capítulo, efectuou-se uma revisão bibliográfica focando os assuntos considerados mais pertinentes no contexto geral dos sistemas de arquitectura paralela e, em particular, da implementação em arquitectura paralela de um algoritmo de síntese de imagens considerado de iluminação global, o *Ray-Tracing*. O objectivo da descrição de implementações paralelas de *ray*-tracing foi reunir, de uma forma concentrada, os melhores trabalhos que, nesse contexto, têm vindo a ser publicados e fazer uma crítica aos mesmos, de forma a orientar as opções tomadas nos capítulos 4 e 5, aquando do desenvolvimento de implementações específicas de *ray-tracing* sobre arquitecturas paralelas.

No capítulo 3 apresentaram-se dois desenvolvimentos realizados com algoritmos de iluminação global, o primeiro dedicado ao problema da reconstrução da função de

iluminação em radiosidade e o segundo ao problema da integração, num mesmo algoritmo de síntese de imagem, de superfícies especulares e difusas.

O algoritmo radiosidade determina a energia por unidade de tempo e de área em locais específicos das superfícies dos objectos que constituem a cena. Para efeitos de visualização, a informação discreta assim obtida tem que ser interpolada, tarefa à qual se dá o nome de reconstrução da função de iluminação.

A interpolação mais vulgarmente utilizada é a bilinear, efectuada em cada *patch*, a partir dos valores de radiosidade nos vértices dos mesmos. Embora produza imagens com aparência suave, só garante uma continuidade de grau C^0 na função de iluminação, pelo que deixa perceber, em alguns casos, o chamado efeito de *Mach band*.

O método desenvolvido é baseado na interpolação bicúbica por superfícies de Hermite e garante uma continuidade de grau C^1 na função de iluminação ao longo das fronteiras dos *patches*. Baseia-se na aproximação da normal à função de iluminação, em cada vértice, pela média aritmética das normais calculadas na vizinhança do vértice, nos *patches* que o partilham. As derivadas parciais e mistas necessárias à definição da matriz de Hermite são calculadas com base na relação entre a normal assim determinada à função de iluminação e a normal geométrica no mesmo ponto.

Os resultados do método apresentam um erro pequeno em termos de valores absolutos, mas não se encontra ainda solucionado o problema do desfasamento encontrado entre as curvas da função de iluminação reconstruída e real. Realmente é visível, principalmente em *patches* de grandes dimensões, um ligeiro deslocamento das áreas de maior brilho, se comparadas com as áreas equivalentes de uma imagem calculada com uma granularidade de *patches* muito fina.

A inclusão, num mesmo algoritmo de síntese de imagem, de superfícies especulares e difusas tem sido tema de investigação nos últimos tempos. Os dois algoritmos de iluminação global, *ray-tracing* e radiosidade gozam, neste aspecto particular de alguma complementaridade. O *ray-tracing* efectua adequadamente o processamento de superfícies especulares, enquanto que o algoritmo radiosidade parte do princípio que todas as superfícies são difusas ideais. O segundo trabalho apresentado no capítulo 3 dedica-se à exploração deste aspecto de complementaridade dos dois algoritmos, de forma a conceber um algoritmo de iluminação global mais geral.

Partindo de um algoritmo de radiosidade por *shooting*, que transfere energia directamente de um *patch* para um vértice, o método desenvolvido determina percursos possíveis para a transferência de energia, nomeadamente por reflexão em *patches* de superfícies especulares puras.

Dado que os percursos gerados por reflexão em *patches* especulares reduzem a energia depositada no receptor, foi necessário redefinir a formulação matemática do cálculo de delta factores de forma de *patch* para vértice. A nova formulação admite várias reflexões sucessivas no mesmo percurso de energia, mas admite-se que, nesse caso, o método dá origem a algumas perdas de energia.

Dado que, por necessidade de direccionar a energia reflectida, os espelhos reflectem imediatamente toda a energia que recebem, o novo método não aumenta o número de iterações de radiosidade, a menos das que são naturalmente necessárias por existirem mais transferências de energia. O incremento no custo computacional do algoritmo relaciona-se, principalmente, com a necessidade de lançar mais raios, um por cada *patch* especular, a partir dos *patches* emissores. No contexto geral de todo o método, o custo computacional não aumenta significativamente.

O *Ray-Tracing* é um algoritmo de iluminação global que permite a síntese de imagens com um aspecto bastante realista. No entanto, a sua relação custo/realismo é demasiado elevada para que possa ser utilizado em ambientes de carácter interactivo.

No capíttulo 4 introduziu-se a noção de Realismo Crescente em *Ray-Tracing*. Genericamente, trata-se de conceder ao utilizador algumas facilidades que lhe permitam acompanhar a geração de uma imagem, de uma forma progressiva, em alternativa a ter que esperrar pelo momento final, em que a imagem se encontra completa. O realismo crescente em *ray-tracing* corresponde a fazer evoluir a qualidade da imagem segundo um conjunto de grandezas, ditas de controlo do realismo crescente. Estas podem ser limitadas pelo utilizador de forma que este seleccione a via mais interessante para fazer progredir a qualidade da imagem.

A demonsstração dessas ideias foi feita na primeira parte do capítulo, onde se apresentou uma verssão sequencial de *ray-tracing* com realismo crescente, obtida a partir de desenvolwimentos anteriores sobre o algoritmo tradicional. Os resultados obtidos são encorajadores, mostrando-se que, mesmo com um esforço computacional bastante inferior ao necessário para criar a imagem final, é possível criar imagens razoavelmente legíveis.

No entantto, a necessidade de acelerar os cálculos, nomeadamente no que se refere às intersecções de raios com objectos, justifica o recurso a arquitecturas de processamento paralelo. Neste sentido, apresenta-se na segunda parte do capítulo uma nova implememtação, sobre uma arquitectura paralela baseada em *Transputers*, destinada a dar continuidade aos testes de realismo crescente e a avaliar as melhores formas de paralelização do problema.

O realismo crescente é obtido por progressividade, em resolução da imagem e em efeitos ópticos traduzidos pela profundidade dos raios nas árvores de iluminação. Assim, um algoritmo deste tipo deve ser dotado da capacidade de processar raios do mesmo nível na árvore de iluminação, em pontos diferentes do ecrã, retomando mais tarde o processamento dos novos raios criados.

No contexto da segunda implementação de *ray-tracing* com realismo crescente, deduziu-se uma nova expressão para o cálculo da intensidade de uma amostra, bastante adaptada aos objectivos em vista. Em vez de percorrer a árvore de iluminação da amostra por um método recursivo, o método implementado, apoiando-se na expressão referida, actualiza a amostra de uma forma incremental, por níveis da árvore. A qualquer momento é possível determinar o efeito que um raio processado tem sobre a amostra respectiva e, também de uma forma incremental, determinar os factores de influência que os seus descendentes terão sobre a mesma amostra.

Os testes efectuados ao método mostram que, com um esforço computacional bastante inferior ao necessário para o cálculo de uma imagem final, um observador treinado e com uma noção da imagem que se encontra em geração pode tomar algumas decisões acerca da colocação dos objectos no espaço e até mesmo de alguns efeitos ópticos. O principal problema detectado é o acentuado efeito de mosaico que as imagens geradas apresentam.

Relativamente à paralelização, efectuada segundo uma *Farm* de processadores, não se atingiram resultados de boa qualidade. A granularidade das tarefas é demasiado fina e a eficiência do sistema diminui com o aumento de processadores, principalmente em cenas de baixa complexidade de cálculo. Outro problema relacionado é a impossibilidade de processar imagens com um número elevado de objectos, devido a limitações de memória local nos vários processadores.

Face aos resultados obtidos no capítulo 4, definiu-se, no capítulo 5, um novo sistema de *Ray-Tracing* com Realismo Crescente Interactivamente controlado pelo utilizador (IIRRA-*Interactive Increasing Realism Ray-tracing Algorithm*).

Como pressupostos iniciais, o novo sistema deveria permitir a distribuição e consequente partilha dos dados pelas memórias locais dos processadores e a partilha de tarefas, cuja gestão passaria a ser local em vez de centralizada. Também deveria reduzir o efeito de mosaico das imagens, assim como explorar a coerência da imagem, de forma que, com o mesmo esforço computacional, permitisse a visualização de uma maior número de detalhes.

Para o efeito, definiu-se uma arquitectura genérica, na qual cada processador contém, além do processo de cálculo de *ray-tracing*, dois outros processos, um de gestão de dados

e outro de gestão de tarefas. Ao longo da rede de processadores, os vários gestores do mesmo tipo comunicam entre si, de forma a partilharem informação, libertando o processo de cálculo de tarefas de baixo nível.

Assim, as comunicações tomam particular importância no sistema e foi necessário desenvolver métodos de *routing* que facilitassem a comunicação de mensagens entre processadores, diminuindo simultaneamente o risco de ocorrência de *deadlock*. Dois métodos conhecidos foram testados e os resultados respectivos apresentados.

Ainda no âmbito dos processos de *routing*, desenvolveu-se um novo tipo de endereçamento, designado por *InPathTo* que pode ser um precioso auxiliar nos acessos a dados remotos que ocorrem por via da partilha de dados e de tarefas. Genericamente, este tipo de endereçamento permite que um item requisitado a um determinado processador possa ser obtido num outro processador mais próximo, diminuindo, consequentemente, a quantidade de mensagens na rede. Um estudo analítico da diminuição da quantidade de mensagens, em função do comprimento total dos percursos e da probabilidade de um pedido ser atendido num processador intermédio, foi efectuado.

Para resolver os problemas de distribuição e partilha de dados, optou-se por uma solução de Memória Partilhada Virtual e criou-se uma hierarquia de acessos a memória. Cada gestor de dados, em cada processador, mantém uma *cache* de dados, segundo uma estratégia LRU que optimiza os acessos que a aplicação local efectua. O espaço na *cache* é gerido ao nível do item de dados em vez de páginas, como é vulgar noutros sistemas do mesmo tipo.

Sempre que um item de dados requisitado não se encontre na *cache*, o gestor de dados envia um pedido de acesso a dados remotos ao processador de coordenação (único que possui a capacidade de endereçar todos os itens de dados), utilizando as facilidades do endereçamento *InPathTo*. A gestão da *cache*, em conjunto com a possibilidade de um determinado item ser obtido antes da mensagem de pedido atingir o processador de coordenação, reduz substancialmente os acessos à rede de comunicações, facto que é de importância fundamental para a eficiência do sistema.

Ainda com a preocupação de diminuir a quantidade de mensagens, foi apresentado um método concebido com o objectivo de diminuir a redundância de mensagens de pedido e de resposta de itens de dados. De acordo com este método, cada gestor de dados determina se um pedido que recebeu de outro processador e ao qual não pode responder, já terá sido pedido por ele próprio. Em caso afirmativo, não reenvia a mensagem para a rede, ficando responsável por responder ao outro processador quando ele próprio receber o item em causa. Em termos de *ray-tracing*, esta estratégia parece ser bastante favorável

nos instantes iniciais de processamento em que os processadores acedem tendencialmente ao mesmo conjunto de objectos.

Os gestores de tarefas foram também alvo de discussão no capítulo 5. É da sua responsabilidade gerir as tarefas, definidas como sendo células imagem, para a aplicação local. A implementação dessa gestão não é directa dado que, contrariamente a outros sistemas de *ray-tracing*, uma tarefa no sistema IIRRA não é processada de uma só vez.

Cada célula imagem tem uma determinada importância em termos de realismo da imagem e, a cada momento, deve ser processada a célula com maior importância, pelo que se coloca um problema de selecção. Por outro lado, a quantidade de tarefas aumenta bastante durante o cálculo, o que cria problemas de espaço em memória (a quantidade de tarefas diminui somente quando o processamento se aproxima do final).

A solução encontrada para os dois problemas mencionados é semelhante à da versão experimental em arquitectura paralela do capítulo 4, ou seja, a classificação das células imagem em células em modo interactivo e células em modo não interactivo, correspondendo às células que num dado momento são consideradas, respectivamente, muito importantes e pouco importantes para a qualidade da imagem. Dos dois grupos, somente o primeiro cede tarefas para processamento, o que permite, por um lado, diminuir a quantidade de memória necessária e, por outro lado, tornar mais eficientes os métodos de selecção das tarefas mais importantes.

Em termos de partilha de tarefas para a realização do balanceamento dinâmico da carga computacional, que é da responsabilidade do processo de gestão de tarefas, o sistema não se encontra ainda finalizado.

Na situação actual, o balanceamento da carga computacional é efectuado de uma forma estática, com base em distribuição de tarefas iniciais tendencialmente equilibradas. Mesmo com um balanceamento dinâmico em funcionamento, o balanceamento *à priori* tem repercussões favoráveis, dado que uma boa distribuição inicial de tarefas diminui a actividade de balanceamento dinâmico e, em consequência, a quantidade de mensagens na rede, já de si bastante sobrecarregada com a partilha de dados.

Quanto às melhorias introduzidas no crescimento do realismo, cada célula imagem é caracterizada por um conjunto de amostras, regularmente distribuídas por níveis de amostragem, de forma a facilitar a sua triangulação e posterior interpolação de cores.

A interpolação de cores tem como objectivo diminuir o efeito de mosaico sentido na versão experimental em arquitectura paralela apresentada no capítulo 4. Os resultados apresentados mostram que este é um avanço razoável nas prestações do sistema.

A regularidade das amostras no interior das células favorece ainda a subdivisão adaptativa destas últimas, sem grandes necessidades de cálculos redundantes. No entanto, alguma redundância existe ainda, pelo que esta é uma via a explorar cuidadosamente em desenvolvimentos futuros.

A subdivisão das células é controlada por um valor limiar de contraste. Assim, torna-se possível manter células de grandes dimensões em zonas da imagem que gozem de bastante coerência ou seja, de fraco contraste, dedicando o esforço computacional em zonas dotadas de maior detalhe. Também neste caso se define uma via para futuros desenvolvimentos.

Realmente, as primeiras imagens criadas possuem células de grandes dimensões, pelo que as amostras se encontram bastante espaçadas. Nestas condições, o valor de contraste medido não é de todo seguro e, por este motivo, ocorrem necessariamente subdivisões indevidas ou, pelo contrário e mais grave, células aparentemente com pouco contraste mas com bastante detalhe no seu interior não são subdivididas.

O método utilizado por J. Leitão *et al.*, em [Lei91], [Lei93], na versão sequencial experimental de *ray-tracing* com realismo crescente apresentada no capítulo 4, garante que uma célula é subdividida se incluir pelo menos um objecto. Se bem que seja um avanço razoável no sentido de garantir a subdivisão de células aparentemente homogéneas, estão ainda por resolver questões como a inclusão de efeitos ópticos numa célula. Um efeito óptico, por exemplo um reflexo, é obviamente um motivo de quebra de homogeneidade, mas a sua detecção é extremamente difícil.

6.2 Desenvolvimentos futuros

No sistema IIRRA, são vários os desenvolvimentos que podem vir a ser efectuados no futuro. O sistema encontra-se ainda em fase de protótipo, com alguns módulos implementados de uma forma incompleta como é o caso da gestão dinâmica do balanceamento de cargas computacionais.

Posteriores desenvolvimentos deverão contemplar o tema do balanceamento dinâmico de cargas computacionais, partindo das soluções gerais publicadas em referências mencionadas no texto e que, pela descrição que efectuam, deixam perceber uma melhor adaptação dos métodos passivos [5.1.3.1] aos requisitos do sistema IIRRA.

Realmente, parece ser razoável a solução de um gestor de tarefas enviar um pedido aos gestores equivalentes de outro ou de outros processadores, quando detecta que o número de tarefas que possui é pequeno. Um dos princípios definidos por S. Green no sistema

DEnIS [Gre91], para efectuar o balanceamento da carga computacional, assenta nesta estratégia. Resta saber a que processadores deve o pedido ser enviado. Várias hipóteses se podem colocar, mas as duas seguintes parecem ser as mais adequadas.

- Um pedido de tarefas é enviado ao processador coordenador, com endereço *InPathTo*, de forma a tentar obter uma resposta no percurso;
- Um pedido é enviado aos processadores que são vizinhos imediatos do processador que efectua o pedido.

A primeira hipótese é algo semelhante à utilizada na distribuição e partilha de dados o sistema. Tem como inconveniente que as mensagens relacionadas com a partilha de tarefas seguem exactamente os mesmos percursos das mensagens que se relacionam com os acessos a dados, sobrecarregando os respectivos *links* dos *transputers*.

A segunda hipótese parece ser mais razoável. Em [Kuc91] descrevem-se alguns métodos de balanceamento dinâmico de cargas computacionais, nesse contexto, que interessa avaliar.

Os gestores de dados em memória partilhada virtual apresentam como principal problema a lentidão de acessos, mesmo quando os itens acedidos se encontram em memória local. Justificou-se o problema, no texto, pelo facto de a implementação dos gestores de dados serem implementados em *software*.

Nos últimos tempos têm sido comercializados novos produtos na área dos *transputers*, baseados em módulos contendo dois processadores, com alguma memória partilhada. Os *transputers* responsabilizam-se pelas comunicações e o outro processador, normalmente de desempenho mais elevado, pela componente de cálculos. Uma melhoria na eficiência do sistema de memória virtual partilhada poderia ser conseguida pela implementação, em processadores diferentes, da aplicação e dos gestores de dados. Seria assim possível que algumas operações dos gestores fossem efectuadas em paralelo com as aplicações.

No entanto, pensa-se que a solução final de máxima eficiência teria sempre que passar por uma implementação dos mesmos em hardware dedicado.

Quanto ao realismo crescente, é necessário incluir o processamento de transparências no sistema. Como foi abordado no texto, assume-se nesta versão intermédia que as transparências não são tratadas, para não aumentar a complexidade das estruturas de dados das células imagem, o que teria efeitos negativos num eventual esquema de partilha dinâmica de tarefas. A abordagem do problema deve portanto ser simultânea com a abordagem do balanceamento dinâmico das cargas computacionais.

Relativamente aos trabalhos apresentados no capítulo 3, sobre o algoritmo radiosidade, novos desenvolvimentos poderiam ser efectuados na área da divisão automática, em *patches*, das várias superfícies, tendo em atenção as linhas de descontinuidade que dizem respeito às fronteiras de penumbra. Estas linhas caracterizam-se por uma continuidade de grau C^0 e devem, em princípio, ser conhecidas *à priori*.

No que respeita ao trabalho de inclusão de superfícies especulares em radiosidade, dever-se-á incidir algum esforço na subdivisão adaptativa dos espelhos em *patches*, tendo em atenção a sua posição relativamente ao resto da cena.

Ficam portanto em aberto, várias pistas de trabalhos muito promissores. Muito esforço de investigação pode (e deve) ainda ser desenvolvido sobre os mesmos, o que se espera venha a acontecer num futuro próximo.

Referências

Referências

[Aci92] ACIERNO, A.; *Using the PRI PAR to Improve Performance of OCCAM2 Programs;* WoTUG newsletter, Vol. 11, Num. 17, Julho de 1992, pp 55-58.

[Aki91] AKIMOTO, T.; MASE, K.; SUENAGA, Y.; *Pixel-Selected Ray-Tracing;* IEEE Computer Graphics & Applications, Vol. 11, Num. 4, Julho de 1991.

[Ama84] AMANATIDES, J.; *RayTracing with Cones;* ACM Computer Graphics (Actas de SIGGRAPH'84), Vol. 18, Num. 3, Julho de 1984, pp 129-136.

[Ama87a] AMANATIDES, John; *Realism in Computer Graphics: A Survey;* IEEE Computer Graphics & Applications, Vol. 7, Num. 1, Janeiro de 1987, pp 44-56.

[Amd67] AMDAHL, G. M.; *Validity of Single-Processor Approach to Achieving Large-Scale Computing Capability* (Actas de AFIPS Conference), Vol. 30, 1967, pp 483-485.

[App68] APPEL, A.; *Some Techniques for Shading Machine Renderings of Solids;* (Actas de Spring Joint Computer Conference), 1968, pp 37-45.

[Arv87] ARVO, J,; KIRK, D.; *Fast Ray Tracing by Ray Classification;* ACM Computer Graphics (Actas de SIGGRAPH'87), Vol. 21, Num. 4, Julho de 1987, pp 55-64.

[Arv88] ARVO, J.; KIRK, D.; *A Survey of Ray Tracing Acceleration Techniques;* SIGGRAPH'88 Course Notes: Introduction to Ray Tracing, 1988.

[Atk89] ATKIN, P.; *Performance Maximization (INMOS TN Nº 17);* Editores INMOS; The Transputer Applications Notebook-Systems and Performance, Trowbridge, Redwood Burn Limited, 1989, pp 280-298.

[Bad90] BADOUEL, D.; BOUATOUCH, K.; PRIOL, T.; *Ray-tracing on Distributed Memory Parallel Computers: strategies for distributing computations and data;* SIGGRAPH'90 Parallel Algorithms and Architecture for 3D Image Generation Course Notes, Dalas, Agosto de 1990.

[Bad90a] BADOUEL, D.; PRIOL, T.; *An Efficient Parallel Ray-tracing Scheme for Highly Parallel Architectures* (Actas de Fifth Eurographics Workshop on Graphics Hardware), Lausane, 1990.

[Bad94] BADOUEL, D.; BOUATOUCH, K.; PRIOL, T.; *Distributing Data and Control for Ray-tracing in Parallel;* IEEE Computer Graphics & Applications, Vol. 14, Num. 4, Julho de 1994, pp 69-77.

[Bas93] BASTOS, Rui M.; BARANOSKI, Gladimir V.; SOUSA, A. Augusto; FERREIRA, F. Nunes; *Uma Alternativa para evitar o Efeito de Mach Band em Radiosidade* (Actas de 5º Encontro Português de Computação Gráfica), Aveiro, 1993.

[Bas93a] BASTOS, Rui M.; SOUSA, A. Augusto; FERREIRA, F. Nunes; *Reconstruction of Ilumination Functions Using Bicubic Hermite Interpolation* (Actas de Fourth EUROGRAPHICS Workshop on Rendering), Paris, 1993.

[Bay73] BAYER, B.; *An Optimum Method for Two-Level Rendition od Continuous-Tone Pictures;* Conference Record of the International Conference on Communications, 1973, pp 26.11-26.15.

[Ber86] BERGMAN, L.; FUCHS, H.; GRANT, E.; SPACH, S.; *Image Rendering by Adaptative Refinement;* ACM Computer Graphics (Actas de SIGGRAPH'86), Vol. 20, Num. 4, Agosto de 1986, pp 29-38.

[Bon92] BONSIEPE, G.; *Teoria e Prática do Design Industrial,* Centro Português de Design, Lisboa, 1992.

[Bou69] BOUKNIGHT, W.; *An Improved Procedure for Generation of Half-tone Computer Graphics Representations;* R-432, University of Illinois, Setembro de 1969.

[Bou80] BOULE, P.; *Etude et Realisation d'Algorithmes pour la Visualisation de Scènes Composées de Facettes Planes;* Tese de Doutoramento, École National Supérieur d'Informatique et de Mathematiques Appliquées de Grenoble, 1980.

[Car84] CARPENTER, L.; *The A-buffer, An Antialised Hidden Surface Method;* ACM Computer Graphics (Actas de SIGGRAPH'84), Vol. 18, Num. 3, Julho de 1984, pp 103-108.

[Cat74] CATMULL, E.; *A Subdivision Algorithm for Computer Display of Curved Surfaces;* Tese de Doutoramento, University of Utah, Salt Lake City, Dezembro de 1974.

[Cha91] CHALMERS, Alan G.; *A Minimum Path System for Parallel Processing;* Tese de Doutoramento, University of Bristol, 1991.

[Cha91a] CHALMERS, A.; PADDON, D.; *Parallel Processing of Progressive Refinement Radiosity Methods* (Actas de Second Eurographics Workshop on Rendering), Barcelona, 1991.

[Cha93] CHALMERS, A.; STUTTARD, D.; PADDON, D.; *Data Management for Parallel Raytracing of Complex Images* (Actas de International Conference on Computer Graphics), Bombay, 1993, pp 149-162.

[Clo65] CLOUGH, R.; TOCHER, J.; *Finit Element Stiffness Matrices for Analysis of the Plate Bending* (Actas de Matrix Methods in Strctural Mechanics), 1965, pp 515-545.

[Coh85] COHEN, M. F.; GREENBERG, D.; *The Hemi-Cube: a Radiosity Solution for Complex Environments;* ACM Computer Graphics (Actas de SIGGRAPH'85), Vol. 19, Num. 3, Julho de 1985, pp 31-40.

[Coh88] COHEN, M. F.; CHEN, S.; WALLACE, J. R.; GREENBERG, D. P.; *A Progressive Refinement Approach to Fast Radiosity Image Generation;* ACM Computer Graphics (Actas de SIGGRAPH'88), Vol. 22, Num. 4, Agosto de 1988.

[Coo84] COOK, R.; PORTER, T.; CARPENTER, L.; *Distributed Ray Tracing;* ACM Computer Graphics (Actas de SIGGRAPH'84), Vol. 18, Num. 3, Julho de 1984, pp 137-146.

[Coo86] COOK, R.; *Stochastic Sampling and Distributed Ray-Tracing;* ACM Transactions on Graphics, Vol. 5, Num. 1, Janeiro de 1986.

[Cos89] COSTA, António C.; SOUSA, A. Augusto; FERREIRA, F. Nunes; *Experiências com Ray-Tracing-Necessidade de Arquitecturas Paralelas* (Actas de 2º Encontro Português de Computação Gráfica), Porto, 1989, pp C1-C11.

[Dei84] DEITEL, H.; *An Introduction to Operating Systems,* Addison-Wesley, 1984.

[Dip84] DIPPÉ, Mark A. Z.; SWENSEN, John; *An Adaptive Subdivision Algorithm and Parallel Architecture for Realistic Image Synthesis;* ACM Computer Graphics (Actas de SIGGRAPH'84), Vol. 18, Num. 3, Julho de 1984, pp 149-158.

[Dip85] DIPPÉ, Mark A. Z.; WOLD, Erling H.; *Antialising Through Stochastic Sampling;* ACM Computer Graphics (Actas de SIGGRAPH'85), Vol. 19, Num. 3, Julho de 1985.

[Far90] FARIN, G.; *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide;* 2ª Edição, Academic Press, 1990.

[Fer90] FERREIRA, F. Nunes; COSTA, A. Cardoso; SOUSA, A. Augusto; BRANCO, V.A.; *3D Graphics Developments and Research at INESC.North;* Computers & Graphics, Vol. 14, 1990, pp 47-53.

[Fer95] FERREIRA, Hugo; MORAIS, João P.; SOUSA, A. Augusto; BASTOS, Rui M.; FERREIRA, F. Nunes; *Uma Alternativa para Inclusão de Superfícies Especulares em Radiosidade* (Actas de 7º Encontro Português de Computação Gráfica), Lisboa, 1995.

[Fol90] FOLEY, J.; VAN DAM, A.; FEINER, S.; HUGHES, J.; *Computer Graphics, Principles and Practice;* 2ª Edição, Addison-Wesley, 1990.

[Fuj86] FUJIMOTO, A.; TANAKA, T.; IWATA, K.; *ARTS: Accelerated Ray-Tracing System;* IEEE Computer Graphics & Applications, Vol. 6, Num. 4, Abril de 1986, pp 16-26.

[Gim89] GIMARC C.; MILUTINOVIC, V.; *RISC Principles, Architecture and Design;* Editores MILUTINOVIC, V.; High-Level Language Computer Architecture, North-Holland, 1988, pp 178-224.

[Gla84] GLASSNER, A. S.; *Space Subdivision for Fast Ray Tracing;* IEEE Computer Graphics & Applications, Vol. 4, Num. 10, Outubro de 1984.

[Gla95] GLASSNER, A.; *Principles of Digital Image Synthesis,* Morgan Kaufmann, 1995.

[Gol87] GOLDSMITH, J.; SALMON, J.; *Automatic Creation of Object Hierarchies for Ray-Tracing;* IEEE Computer Graphics & Applications, Vol. 7, Num. 5, Maio de 1987, pp 14-20.

[Gor84] GORAL, C. M.; TORRANCE, K. E.; GREENBERG, D. P.; BATTAILE, B; *Modeling the Interaction of Light Between Diffuse Surfaces;* ACM Computer Graphics (Actas de SIGGRAPH'84), Vol. 18, Num. 3, Julho de 1984, pp 213-222.

[Gou71] GOURAUD, H.; *Continuous Shading of Curved Surfaces;* IEEE Transactions on Computers, Vol. C-20, Num. 6, Junho de 1971, pp 623-629.

[Gre89] GREEN, Stuart; PADDON, Derek J.; *Exploiting Coherence for Multiprocessor Ray-Tracing;* IEEE Computer Graphics & Applications, Vol. 9, Num. 6, Novembro de 1989, pp 12-26.

[Gre89a] GREEN, Stuart; PADDON, Derek J.; *A Highly Flexible Multiprocessor Solution for Ray-Tracing;* The Visual Computer, Vol. 5, Num. 6, Dezembro de 1989.

[Gre91] GREEN, Stuart; *Parallel Processing for Computer Graphics,* PITMAN, 1991.

[Gus88] GUSTAFSON, J. L.; *Reevaluating Amdahl's Law;* Comunications of ACM, Vol. 31, Num. 5, Maio de 1988, pp 532-533.

[Hai86] HAINES, E.; GREENBERG, D.; *The Light Buffer: A Shadow-Testing Accelerator;* IEEE Computer Graphics & Applications, Vol. 6, Num. 9, Setembro de 1986, pp 6-16.

[Hai87] HAINES, E.; *A Proposal for Standard Graphics Environments;* IEEE Computer Graphics & Applications, Vol. 7, Num. 11, Novembro de 1987, pp 3-5.

[Hai91] HAINES, E.; WALLACE, J.; *Shaft Culling for Efficient Ray-Traced Radiosity* (Actas de Second Workshop on Rendering), Barcelona, 1991.

[Hal83] HALL, R; GREENBERG, D.; *A Testbed for Realistic Image Synthesis;* IEEE Computer Graphics & Applications, Vol. 3, Num. 8, Novembro de 1983, pp 10-20.

[Hay88] HAYES, John P.; *Computer Architecture and Organization;* 2ª Edição, McGraw-Hill, 1988.

[Hec84] HECKBERT, P.; HANRAHAN; *Beam Tracing Polygonal Objects;* ACM Computer Graphics (Actas de SIGGRAPH'84), Vol. 18, Num. 3, Julho de 1984, pp 119-128.

[Hoc81] HOCKNEY, R.W; JESSHOPE, C. R.; *Parallel Computers,* Adam Hilger Ltd, Bristol, 1981.

[Hwa93] HWANG, Kai; *Advanced Computer Architecture: Parallelism, Scalability, Programmability,* McGraw-Hill, 1993.

[Imm86] IMMEL, D.; COHEN, M.; GREENBERG, D.; *A Radiosity Method for Non-Difuse Environments;* ACM Computer Graphics (Actas de SIGGRAPH'86), Vol. 20, Num. 4, Agosto de 1986, pp 133-142.

[Inm88a] INMOS; *Transputer Instruction Set: A Compiler Writer's Guide,* Prentice Hall, 1988.

[Inm88b] INMOS; *OCCAM2 Reference Manual,* Prentice Hall, 1988.

[Inm92] INMOS; *The Transputer Databook;* 3ª Edição, SGS-Thomson, 1992.

[Isl91] ISLER, Veysi; AYKANAT, Cevdet; ÖZGÜÇ, Bülent; *Subdivision of 3D Space Based on the Graph Partitioning for Parallel Ray-Tracing* (Actas de Second Workshop on Rendering), Barcelona, 1991.

[Jan83] JANSEN, F; WIJK, J.; *Fast Previewing Techniques in Raster Graphics;* Editores ten Hagen, P. (Actas de EUROGRAPHICS'83, Zagreb,1983), North Holland, 1983, pp 195-202.

[Jev89] JEVANS, D.; *Optimistic Multiprocessor Ray-Tracing;* Editores EARNSHAW, R; WYVILL, B.; New Advances in Computer Graphics (Actas de CG International'89), Springer Verlag, 1989, pp 507-522.

[Jon87] JONES, G.; *Programming in OCCAM,* Prentice-Hall, 1987.

[Jon89] JONES, G.; *Carefully Scheduled Selection with ALT;* OCCAM User Group Newsletter, Num. 10, Janeiro de 1989, pp 17-23.

[Kaj86] KAJIYA, J.J.; *The Rendering Equation;* ACM Computer Graphics (Actas de SIGGRAPH'86), Vol. 20, Num. 4, Agosto de 1986, pp 143-150.

[Kay86] KAY, Timothy; KAJIYA, J.J.; *Ray Tracing Complex Scenes;* ACM Computer Graphics (Actas de SIGGRAPH'86), Vol. 20, Num. 4, Agosto de 1986, pp 269-278.

[Kir90] KIRK, D.; VOORHIES, D.; *The Rendering Architecture of the DN10000VS;* ACM Computer Graphics (Actas de SIGGRAPH'90), Vol. 24, Num. 4, Agosto de 1990, pp 299-307.

[Knu73] KNUTH, D. E.; *The art of Computer Programming: Sorting and Searching,* Vol. 3, Addison Wesley, 1973.

[Kob89] KOBAYASHI, H.; HORIGUCHI, S.; KUBOTA, H.; NAKAMURA, T.; *Efective Parallel Processing for Synthesizing Continuous Images;* Editores EARNSHAW, R; WYVILL, B.; New Advances in Computer Graphics (Actas de CG International'89), Springer Verlag, 1989, pp 343-352.

[Kuc91] KUCHEN, H.; WAGENER, A.; *Comparison of Dynamyc Load Balancing Strategies,* Elsevier Sciences Publisher, North-Holland, 1991, pp 303-314.

[Lee85] LEE, Mark E.; REDNER, Richard A.; USELTON, Samuel P.; *Statisticaly Optimized Sampling for Distributed Ray-Tracing;* ACM Computer Graphics (Actas de SIGGRAPH'85), Vol. 19, Num. 3, Julho de 1985.

[Lef93] LEFER, Wilfrid; *An Efficient Parallel Ray Tracing Scheme for Distributed Memory Parallel Computers;* Editores Steve Cunningham (Actas de 1993 Parallel Rendering Symposium), San Jose, California, 1993, pp 77-80.

[Lei91] LEITÃO, J. Miguel; SOUSA, A. Augusto; COSTA, A. Cardoso; FERREIRA, F. Nunes; *Estratégias e Implementação de Realismo Crescente em Ray-Tracing* (Actas de 4º Encontro Português de Computação Gráfica), Lisboa, 1991, pp 299-319.

[Lei93] LEITÃO, J. Miguel; SOUSA, A. Augusto; COSTA, A. Cardoso; FERREIRA, F. Nunes; *Strategies and Implementation of Ray-Tracing with Increasing Realism;* Graphics Modeling and Visualization in Science & Technology (Actas de Workshop on Graphics Modeling and Visualization in Science an Technology, Darmstadt, April 92), Springer-Verlag, 1993.

[Mai92] MAILLOT, J. L.; CARRARO, L; PEROCHE, B.; *A Progressive Ray-Tracing;* Editores CHALMERS, A.; PADDON, D. (Actas de Third Workshop on Rendering), Bristol, 1992.

[May89] MAY, D.; SHEPHERD, R.; *Communicating Process Computers (INMOS TN Nº 22);* Editores INMOS; The Transputer Applications Notebook-Architecture and Software, Trowbridge, Redwood Burn Limited, 1989, pp 33-46.

[May89b] MAY, D.; SHEPHERD, R.; *The Transputer Implementation of OCCAM (INMOS TN Nº 21);* Editores INMOS; The Transputer Applications Notebook-Architecture and Software, Trowbridge, Redwood Burn Limited, 1989, pp 21-32.

[May89c] MAY, D.; KEANE, C.; *Compiling OCCAM into Silicon (INMOS TN Nº 23);* Editores INMOS; The Transputer Applications Notebook-Architecture and Software, Trowbridge, Redwood Burn Limited, 1989, pp 47-59.

[Mit87] MITCHEL Don P.; *Generating Antialiased Images at Low Sampling Densities;* ACM Computer Graphics (Actas de SIGGRAPH'87), Vol. 21, Num. 4, Julho de 1987, pp 65-72.

[Mur90] MURAKAMI, K; HIROTA, K.; *Incremental Ray-Tracing* (Actas de Workshop on Photosimulation, Realism and Physics in Computer Graphics), 1990.

[Nem86] NEMOTO, K; OMACHI, T.; *An adaptative Subdivision by Sliding Boundary Surfaces for Fast Ray-Tracing* (Actas de Graphics Interface'86), 1986, pp 43-48.

[New72] NEWELL, M.; NEWELL, R,; SANCHA, T; *A New Approach to the Shaded Picture Problem;* ACM National Conference, 1972.

[Pac89] PACKER, J.; *Exploiting Concurrency: a Ray-Tracing Example (INMOS TN Nº 7);* Editores INMOS; The Transputer Applications Notebook-Architecture and Software, Trowbridge, Redwood Burn Limited, 1989, pp 144-154.

[Pat92] PATTANAIK, S. N.; MUDUR, S.P.; *Computation of Global Ilumination by Monte Carlo Simulation of the Particle Model of Light;* (Actas de Third Eurographics Workshop on Rendering), Maio de 1992, pp 71-83.

[Pho75] PHONG, Bui-Tuong; *Ilumination for Computer Generated Pictures;* Comunications of ACM, Vol. 18, Num. 6, Junho de 1975, pp 311-317.

[Por79] PORTER, T.; *How Architects Visualize,* Studio Vista-Macmillan Publishing, New York, 1979.

[Pou87] POUNTAIN, D.; MAY, D.; *A Tutorial Introduction to OCCAM Programming,* BSP Professional Books, 1987.

[Pow77] POWEL, M.; SABIN M.; *Piecewise Quadratic Aproximation on Triangles;* ACM Transactions on Mathematical Software, Dezembro de 1977, pp 316-325.

[Pur90] PURGATHOFER, Werner; ZEILLER, Michael; *Configuring Transputers for Ray-Tracing;* Editores FREEMAN, Len; PHILLIPS, Chris; Applications of Transputers 1 (Actas de First International Conference on Applications of Transputers, Liverpool, Aug. 1989), Amsterdam, IOS Press, 1990.

[Qua92] QUARANTE, D.; *Disegño Industrial - Elementos Teóricos,* Vol. 2, Editiones CEAC, Barcelona, 1992.

[Qui94] QUINN, M. J.; *Parallel Computing, Theory and Practice,* McGraw-Hill, 1994.

[Rei95] REINHARD, E.; ZWAAN, M.; JANSEN, F.; *Pyramid Clipping for Efficient Ray Traversal;* Editores HANRAHAN, P.; PURGATHOFER, W. (Actas de Sixth Workshop on Rendering), Dudlin, (para publicação em) Springer-Verlag, 1995.

[Ris94] RIS, Philippe; ARQUÈS, Didier; *Parallel Ray Tracing Based upon a Multilevel Topological Knowledge Acquisition of the Scene;* Computer Graphics Forum (Actas de EUROGRAPHICS'94), Vol. 13, Num. 3, Setembro de 1994, pp C221-C232.

[Rom70] ROMNEY, G. W.; *Computer Assisted Assembly and Rendering of Solids;* TR-4-20, University of Utah, 1970.

[Sal92] SALESIN, D.; LISCHINSKI, D.; DeROSE, T.; *Reconstructing Illumination Functions with Selected Discontinuities;* Editores CHALMERS, A; PADDON, D. (Actas de Third Workshop on Rendering), Bristol, 1992, pp 99-112.

[Sch88] SCHWEDERSKI, T.; MEYER, D.; SIEGEL, H; *Parallel Processing;* Editores MILUTINOVIC, V.; Computer Architecture, Concepts and Systems, North-Holland, 1988, pp 178-224.

[Seq89] SÉQUIN, Carlo H.; SMYRL, Eliot K.; *Parameterized Ray-Tracing;* ACM Computer Graphics (Actas de SIGGRAPH'89), Vol. 23, Num. 3, Julho de 1989.

[Shi87] SHINYA, M.; TAKAHASHI, T.; NAITO, S.; *Principles and Applications of Pencil Tracing;* ACM Computer Graphics (Actas de SIGGRAPH'87), Vol. 21, Num. 4, Julho de 1987, pp 45-54.

[Sil88] SILBERSCHATZ, A.; PETERSON, J.; *Operating Systems Concepts;* 2ª Edição, Addison-Wesley, 1988.

[Sil89] SILLION, F.; PUECH, C.; *A General Two-pass Method Integrating Specular and Difuse Reflection;* ACM Computer Graphics (Actas de SIGGRAPH'89), Vol. 23, Num. 3, Julho de 1989.

[Sou87] SOUSA, A. Augusto; *Cálculo de Visibilidade em Cenas 3D: Síntese de Conceitos, Implementação de um Algoritmo;* Provas de Aptidão Pedagógica e Capacidade Científica, Faculdade de Engenharia da Universidade do Porto, Maio de 1987.

[Sou88] SOUSA, A. Augusto; FERREIRA, F. Nunes; *Síntese de Imagens com Elevado Nível de Realismo* (Actas de 1º Encontro Português de Computação Gráfica), Lisboa, 1988, pp 84-96.

[Sou90] SOUSA, A. Augusto; COSTA, A. Cardoso; FERREIRA, F. Nunes; *Interactive Ray-Tracing for Image Production with Increasing Realism;* Editores VANDONI, C.E.; DUCE, D.A. (Actas de EUROGRAPHICS'90, Montreux, 1990), Vol. 3, Amsterdam, North-Holland, 1990, pp 449-457.

[Sou90a] SOUSA, A. Augusto; COSTA, A. Cardoso; LEITÃO, J. Miguel; FERREIRA, F. Nunes; *Ray-Tracing Controlável Interactivamente para Produção de Imagens de Realismo Crescente* (Actas de 3º Encontro Português de Computação Gráfica), Coimbra, 1990.

[Sou92] SOUSA, A. Augusto; FERREIRA, F. Nunes; *Relatório Técnico JNICT, PMCT/C/TIT/484.90; IIRRA Versão 1: Análise de Resultados,* Porto, Outubro de 1992.

[Sou95] SOUSA, A. Augusto; FERREIRA, F. Nunes; *A Parallel Implementation of an Interactive Ray-Tracing Algorithm;* (para publicação em) Computing Systems in Engineering (Actas de 1st International Meeting on Vector and Parallel Processing, Porto, Setembro de 1993.), Elsevier Science Ltd, 1995.

[Sou95a] SOUSA, A. Augusto; FERREIRA, F. Nunes; *On Writing a Router for Message Passing in a Transputer Network;* (para publicação em) Computing Systems in Engineering (Actas de 1st International Meeting on Vector and Parallel Processing, Porto, Setembro de 1993.), 1995.

[Sun92] SUNG, K.; *The Area Sampling Machine;* Editores CHALMERS, A.; PADDON, D. (Actas de Third Eurographics Workshop on Rendering), Bristol, 1992, pp 147-160.

[Sun92a] SUNG, K.; *Area Sampling Buffer: Tracing Rays with Z-Buffer Hardware;* Computer Graphics Forum (Actas de EUROGRAPHICS'92), Vol. 11, Num. 3, Setembro de 1992, pp C299-C310.

[Sun93] SUN, X.; NI, L.; *Scalable Problems and Memory-Bounded Speedup;* Journal of Parallel and Distributed Computing, Num. 19, 1993, pp 27-37.

[Sut74] SUTHERLAND, I.; SPROULL, R.; SHUMACKER, R.; *A Characterization of Ten Hidden-Surface Algorithms;* Computing Surveys, Vol. 6, Num. 1, Março de 1974.

[Tan90] TANAKA, T.; TAKAHASHI, T.; *Cross Scanline Algorithm:* Editores VANDONI, C. E.; DUCE, D. A.; EUROGRAPHICS'90, Montreux, North-Holland, 1990, pp 63-74.

[Tan94] TANAKA, T.; TAKAHASHI, T.; *Cross Scan Buffer and its Applications;* Editores DÆHLEN, M.; KJELLDAHL, L.; EUROGRAPHICS'94, Oslo, North-Holland, 1994, pp C467-C476.

[Tar83] TARJAN, R.; *Data Structures and Network Algorithms,* Society for Industrial and Applied Mathematics, 1983.

[Tas91] TASTL, Ingeborg; PURGATHOFER, Werner; *Color Spaces and Human Color Perception* (Actas de Second Workshop on Rendering), Barcelona, 1991.

[Wal48] WALD, A.; *Sequential Analysis,* Willey & Sons, 1948.

[Wal87] WALLACE, J. R.; COHEN, M. F.; GREENBERG, D. P.; *A Two-pass Solution to the Rendering Equation: A Synthesis of Ray-Tracing and Radiosity Methods;* ACM Computer Graphics (Actas de SIGGRAPH'87), Vol. 21, Num. 4, Julho de 1987, pp 311-320.

[Wal89] WALLACE, J. R.; ELMQUIST, Kells A.; HAINES, Eric; *A Ray-Tracing Algorithm for Progressive Radiosity;* ACM Computer Graphics (Actas de SIGGRAPH'89), Vol. 23, Num. 3, Julho de 1989.

[Wat70] WATKINS, G.; *A Real Time Visible Surface Algorithm;* UTECH-CSC-70-101, University of Utah, Julho de 1070.

[Wat92] WATT, Alan; WATT, Mark; *Advanced Animation and Rendering Techniques, Theory and Practice,* Addison-Wesley, 1992.

[Wei77] WEILER, K; ATHERTON, P.; *Hidden Surface Removal Using Polygon Area Sorting;* ACM Computer Graphics (Actas de SIGGRAPH'77), Vol. 11, Num. 2, Julho de 1977, pp 214-222.

[Whi80] WHITTED, Turner; *An Improved Illumination Model for Shaded Display;* Comunications of ACM, Vol. 23, Num. 6, Junho de 1980, pp 343-349.

[Whi92] WHITMAN, S.; *Multiprocessor Methods for Computer Graphics Rendering,* Jones and Bartlett Publishers, 1992.

[Wil78] WILLIAMS, L.; *Casting Curved Shadows on Curved Surfaces;* ACM Computer Graphics (Actas de SIGGRAPH'78), Vol. 12, Num. 3, Agosto de 1978, pp 270-274.

[Won87] WONG, S.; CENDES, Z.; *C1 Quadratic Interpolation Over Arbitrary Point Sets;* IEEE Computer Graphics & Applications, Vol. 7, Num. 11, Novembro de 1987, pp 8-17.

Secção de Cores

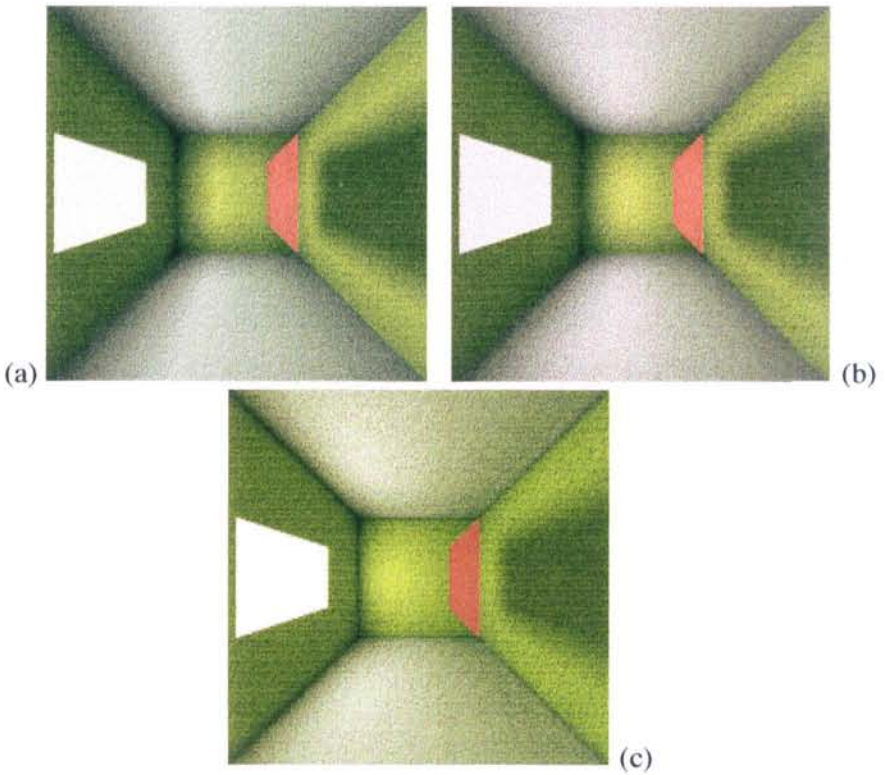


Fig. 3.8 - Imagens calculadas com interpolação bilinear (a) e bicúbica (b) e imagem de referência (c)

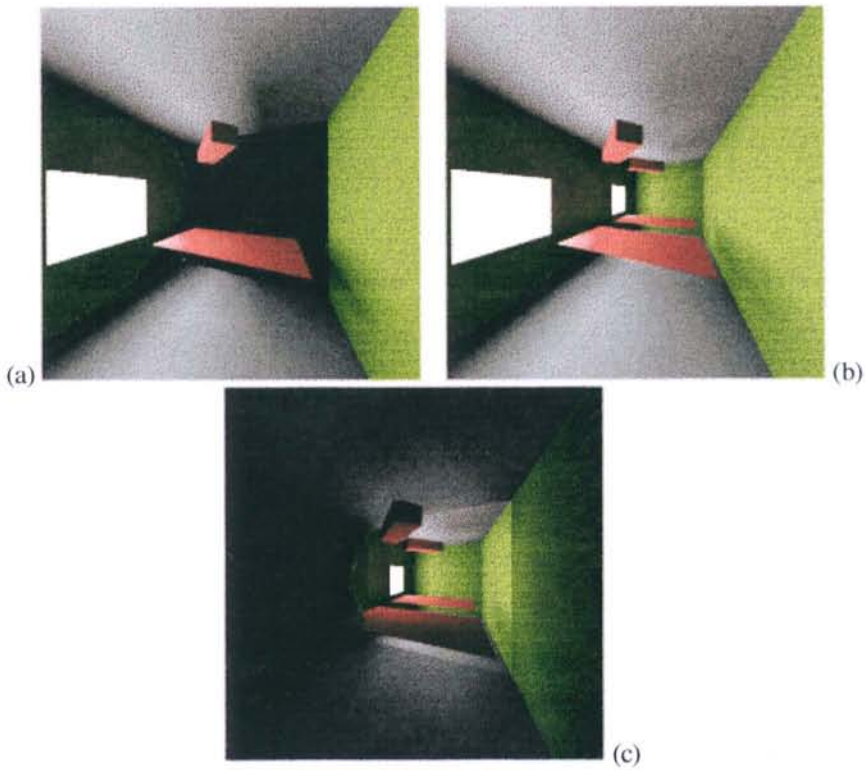


Fig. 3.16 - Imagens de uma cena completamente difusa (a), com um espelho (b) e diferença das duas (c)

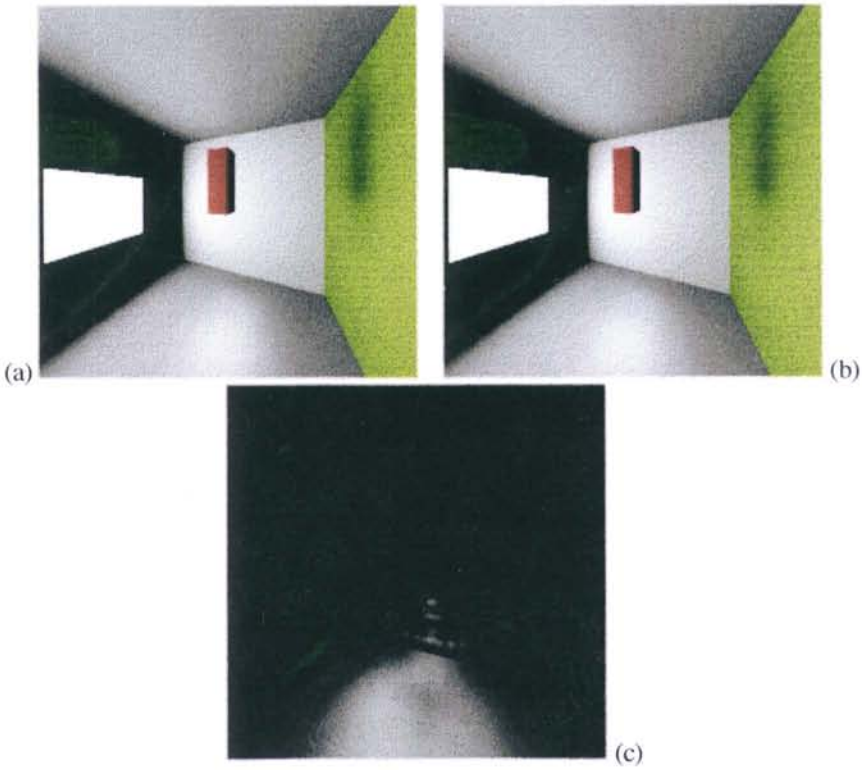
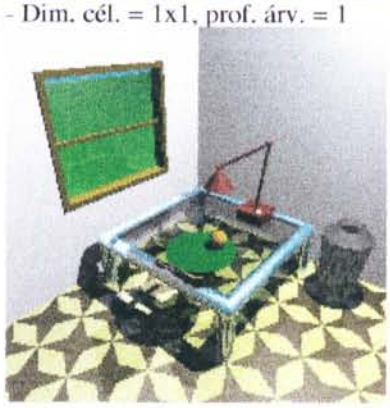


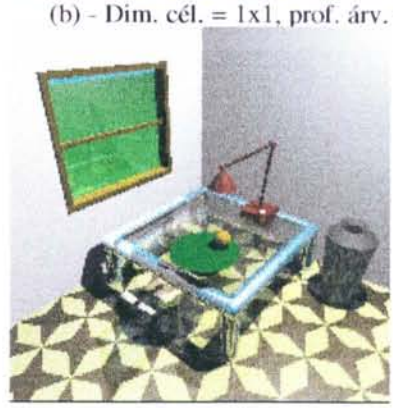
Fig. 3.18 - Imagens de uma cena iluminada por um foco difuso (a), um foco especular (b) e diferença escalada das duas (c)

(a) - Dim. cél. = 1x1, prof. árv. = 1

(b) - Dim. cél. = 1x1, prof. árv. = 2



(c) - Dim. cél. = 1x1, prof. árv. = 3



(d) - Dim. cél. = 1x1, prof. árv. = n

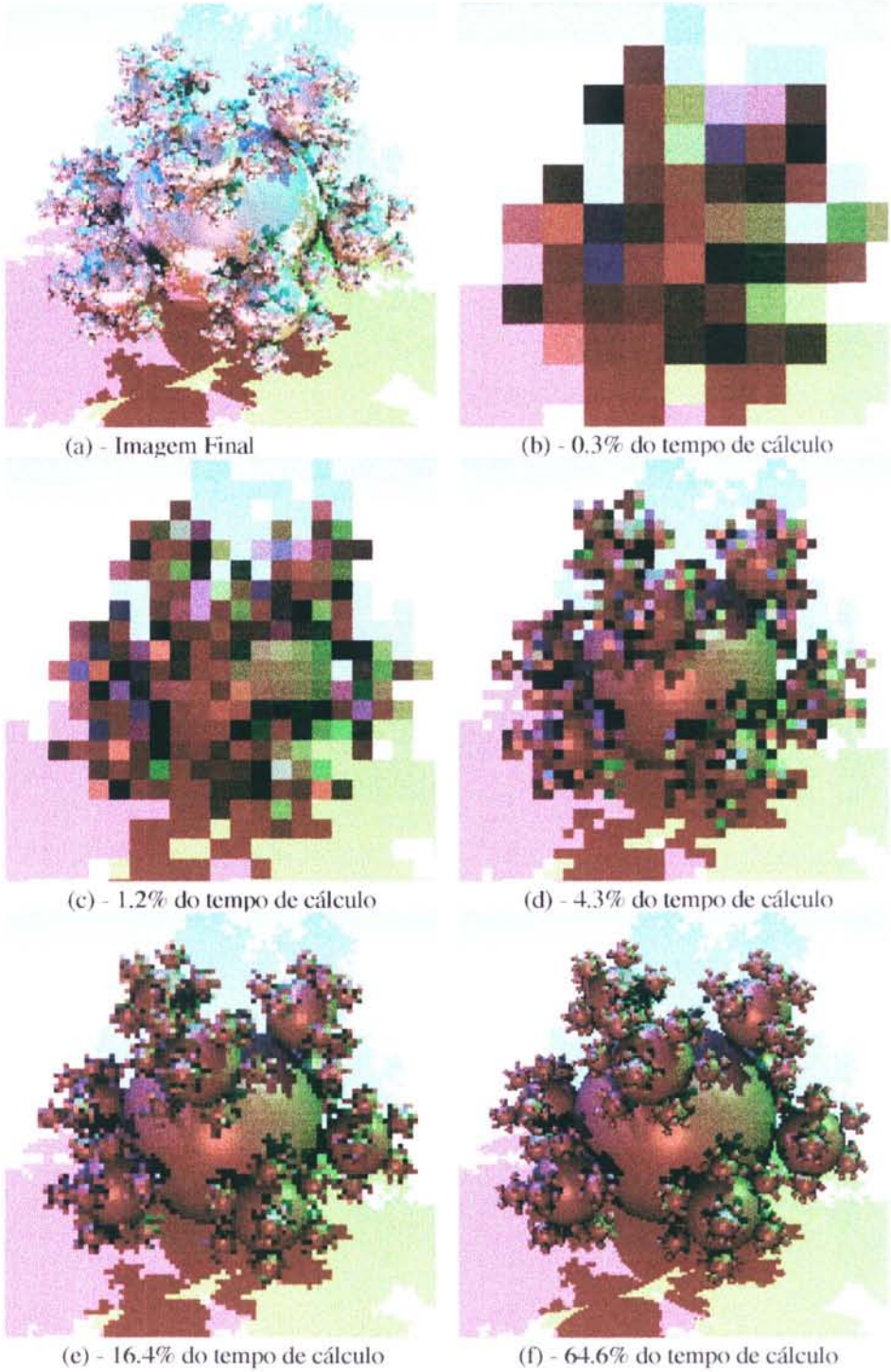
Fig. 4.5- Sequência de imagens obtida por incrementos na profundidade das árvores de iluminação

(a) - Dim. cél. = 4x4, contraste baixo

(b) - Dim. cél. = 4x4, contraste elevado

(c) - Dim. cél. = 2x2, contraste elevado

Fig. 4.6 - Sequência de imagens obtida por progressão em resolução e em contraste das células imagem



(a) - Imagem Final

(b) - 0.3% do tempo de cálculo

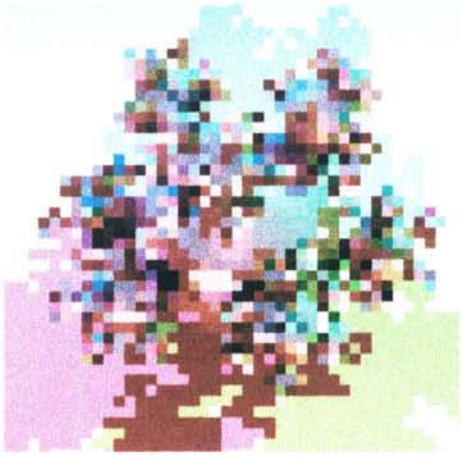
(c) - 1.2% do tempo de cálculo

(d) - 4.3% do tempo de cálculo

(e) - 16.4% do tempo de cálculo

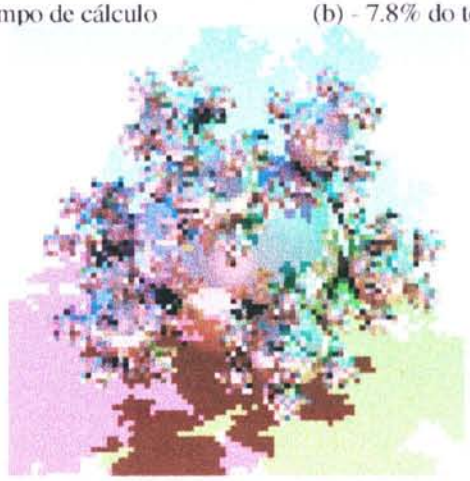
(f) - 64.6% do tempo de cálculo

Fig. 4.18- Imagem final e uma sequência de realismo crescente por progressão em resolução



(a) - 5.1% do tempo de cálculo

(b) - 7.8% do tempo de cálculo



(c) - 23.5% do tempo de cálculo

Fig. 4.19- Sequência de realismo crescente por progressão em profundidade na árvore de iluminação e em resolução

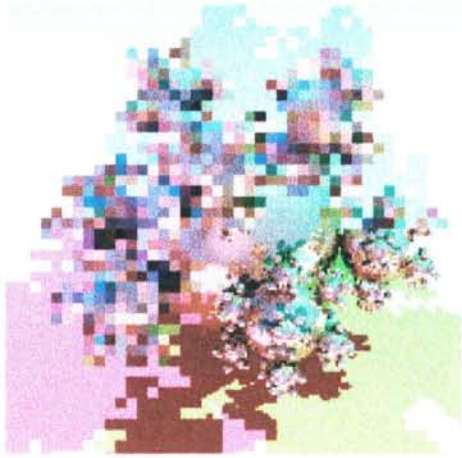


Fig. 4.20 - Definição de uma área de interesse