Análise de Desempenho no Processamento de Dados Geográficos Irregularmente Distribuídos, Provenientes de um Sensor LIDAR

Péricles Filomeno Monteiro Pinto

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Informática
Rua Roberto Frias, s/n, 4200-465 Porto, Portugal

Maio de 2008

Análise do Desempenho no Processamento de Dados Geográficos Irregularmente Distribuídos, Provenientes de um Sensor LIDAR

Péricles Filomeno Monteiro Pinto

Trabalho submetido para obtenção do grau de Mestre em Engenharia Informática e Computação no âmbito do processo Bolonha, na Faculdade de Engenharia da Universidade do Porto.

Orientadores:
Prof. Doutor Luís Paulo Reis
Prof. Doutor Luís Gonçalves Seco

Maio de 2008

Resumo

O varrimento aéreo por laser tornou-se numa tecnologia corrente na aquisição rápida de dados topográficos e cartografia da superfície terrestre. O resultado de um voo LiDAR é uma densa nuvem irregular 3-D de pontos (X, Y, Z) que descreve a forma da superfície num determinado momento, ou seja, os pontos que definem o solo e os que definem os objectos que aparecem sobrepostos

Actualmente, diversos algoritmos de filtragem foram já desenvolvidos para a derivação de Modelos Digitais do Terreno (MDT). No entanto, a maior parte dos estudos comparativos sobre questões de performance entre diferentes aproximações centram-se na eficácia da filtragem propriamente dita não contemplando questões relacionadas com o custo computacional que consideramos fundamental devido à quantidade de dados que são necessários analisar e processar.

Se tivermos em conta que a maior parte dos algoritmos que foram desenvolvidos são iterativos e aplicados directamente sobre a nuvem irregular de pontos, temos um problema acrescido dado que essas relações de proximidade só são possíveis através de um cálculo prévio das distâncias euclidianas entre todos os pontos, que dependendo da quantidade de dados poderá traduzir-se num elevado tempo de processamento.

Nesta investigação realizamos uma avaliação do custo computacional estimado em tempo de processamento de um Filtro Morfológico Adaptativo (FMA) manipulando dados em disco optimizados sobre dois métodos de indexação espacial e comparamos a sua evolução em dimensão de dados com o mesmo método baseado em memória RAM. A partir do FMA, foi proposta uma função de optimização de forma a reduzir o número de pesquisas espaciais e foram incorporados dois métodos de indexação com dados armazenados em disco no filtro optimizado, baseado na R-Tree e QuadTree de forma avaliar o seu desempenho.

Mediante a dimensão dos dados analisados, constatou-se que a QuadTree apresenta ligeiramente melhores resultados, no entanto notou-se uma tendência para a R-Tree ser melhor á medida que a dimensão dos dados aumenta.

Os resultados obtidos, demonstram que a função de optimização executada em memória, a partir de 200.000 pontos consegue obter ganhos de tempo na ordem dos 89 %, relativamente ao convencional. No entanto, a quantidade de dados que consegue filtrar está limitado à quantidade de memória RAM disponível da máquina que é executada.

Constatou-se que apesar da solução original não optimizada estar a executar em memória, a solução implementada com acessos ao disco começa a apresentar melhor resultado no intervalo de] 50; 200.000 [pontos. Verificou-se ainda que de 2500 para 200000 pontos a solução original não optimizada aumenta cerca de 11000 vezes, na optimizada em memória 200 vezes e nas optimizações em disco 100 vezes.

Abstract

Aerial laser scanning has become a current technology in the rapid acquisition of topographic data and terrestrial surface cartography. The result of a LIDAR (*Light Detection and Ranging*) flight is a dense, irregular cloud of 3D points (X,Y,Z) that describes the surface at a given time.

Several algorithms have already been developed for deriving Digital Terrain Models (DTM). However, most of the comparative studies about performance between distinct approaches are centered in the filtering efficacy and not in questions regarding the computational cost that is fundamental due to the amount of data that is needed to process.

If it is taken into account that most of the algorithms developed are iterative method applied directly over the irregular cloud of points, we have an increased problem, since these proximity relations are only possible through a previous calculation of the Euclidian distances among all the points. This process, depending on the amount of data may take a very long processing time.

In this research project the computational cost and estimated processing time of an adaptative morphological filter (ADF) was analyzed. The process used was to manipulate the optimized data in disk using two spatial indexing methods and comparing its evolution with the same method based in RAM memory. An optimization function was proposed reducing the number of spatial queries and incorporating the two indexing methods, based on R-Tree and QuadTree, with data stored in disk in the optimized filter, evaluating its performance.

It was concluded that the QuadTree achieves slightly better results in the tests performed. However, it was noted a tendency for the R-Tree method to improve comparatively when the amount of data increases.

The results achieved show that the optimization function, executed in memory, for sets of over 200.000 points, achieves a gain in the computational time of about 89%, relatively to the conventional process. However, the amount of data that it achieves to filter is limited to the amount of RAM available in the machine where the method is executed.

It was also concluded that although the original non optimized solution executes in memory, the implemented solution, even using disk accesses, starts to achieve better results in the] 50; 200.000[points interval. It was also verified that from 2500 to

200000 points, the original, non optimized solution, increases by about 11000 the compuational time, while the optimized in memory increases 200 times and the optimized in disk increases the time by only 100 times.

Aos Meus Pais

Agradecimentos

Agradeço às pessoas e entidades que de alguma forma contribuíram para a realização desta dissertação, com especial destaque aos meus orientadores Luís Gonçalves Seco e Luís Paulo Reis, pela oportunidade que me deram em realizar esta investigação e por sempre me ajudarem a ordenar e centrar as ideias no caminho certo.

Ao professor José Manuel Matos Moreira do Departamento Electrónica Universidade de Aveiro, pelos seus conselhos e pela disponibilidade demonstrada que se traduziram em bons ensinamento para este trabalho.

Ao professor Jorge Manuel Gomes Barbosa, Departamento de Informática, Faculdade de Engenharia, da Universidade do Porto, pelos esclarecimentos que foram importantes nesta investigação.

Ao *instituto Galego de estadística* que suportou a aquisição dos dados LIDAR, componente indispensável para a realização desta dissertação.

Ao centro de investigação e Ciências Geo-Espaciais, das Faculdade Ciências, Universidade do Porto, pelo fornecimento de todos os meios necessários para a realização deste trabalho.

À minha família pelo apoio incondicional que sempre me deram.

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1

1. Introdução

"*Building the database is usually the most time consuming, sometimes frustrating part of GIS work, no matter what kind of software you use. However, once the data are gathered and nicely integrated in a consistent manner, the work with geospatial data may become quite exciting and rewarding*." (Neteler and Mitasova 2003).

1.1 Enquadramento e Motivação

Uma gestão sustentada do espaço onde vivemos é crucial desde o ponto de vista económico, social e ecológico. A concretização desse objectivo tem de passar impreterivelmente por ter à disposição informação de qualidade sobre o território.

Essa gestão passa pela utilização de Sistemas de Informação Geográfica[1] (SIG) (Worboys & Duckham; Longley et al., 2005), onde a exactidão e funcionalidade da informação introduzida nos SIG dependem em grande medida da qualidade dos dados topográficos (Chrisman, 1984; Borges, 1977) e da sua velocidade de aquisição. Essa compilação consome uma parte importante dos recursos dos projectos em termos de tempo e custo, assumindo assim, uma importância considerável que se vê incrementada dia a dia, pela necessidade de inventários cada vez mais complexos com um tempo de captura reduzido.

[1] São sistemas computacionais capazes de capturar, armazenar, consultar, manipular, analisar e imprimir dados referenciados espacialmente à superfície da terra.

Nos últimos 30 anos, com o desenvolvimento mais acentuado das Tecnologias de Informação e Comunicação (TIC), essa evolução foi preponderante nas áreas da Detecção Remota[2] (Fonseca & Fernandes, 2003; Aronoff, 2005) que têm vindo a revolucionar a forma como são realizados os levantamentos de dados geográficos.

Neste domínio, salienta-se a tecnologia de varrimento aéreo por laser comummente conhecida como LiDAR (*Light Detection And Ranging*) ou *Airborne Laser Scanning* (ALS) que tem evoluído de forma notável (Ackerman, 1999), convertendo-se actualmente num instrumento eficiente e de baixo custo (Petzold *et al*., 1999; Fowler, 2001; Flood, 2001).

O seu desenvolvimento deu-se no contexto da modelação da superfície terrestre, para a criação de Modelos Digitais de Superfície[3] (MDS) e Modelos Digitais do Terreno[4] (MDT) (Sithole, 2005), no entanto, tem-se estendido em áreas que vão desde a estimação de variáveis relacionadas com a planificação e gestão florestal, entre as quais, dendométricas (Næsset *et al.*, 2004; Gonçalves-Seco *et al*., 2007a; Hyyppä *et al*., 2008), relacionadas com o risco de incêndio florestal (Riaño *et al.*, 2003; Riaño *et al*., 2007; Gonçalves-Seco *et al*., 2007b) e os *stock*s actuais de biomassa (Lefsky *et al*., 1999; Riaño *et al*., 2003; Hall *et al*., 2004; Gonzalez *et al*., 2005) até classificação de objectos (Song et al., 2002; Arefi et al., 2003; Charaniya *et al.*, 2004; Brennan et al., 2006; Gonçalves & Gonçalves-Seco, 2007) e produção de modelos 3D (Ameri, 2000; McIntosh, 2000; Vosselman e S. Dijkman, 2002; Zhang et al., 2007), estudos especificos de telecomunicações (Vogtle, 2000), de corredores lineares (Clode & Rottensteiner, 2005), planeamento e construções de estradas (Gomes Pereira & Janssen, 1999), monitorização de diques e da zona costeira (Vassen et al., 1998; Irish et al., 1999), monitorização de zonas de cheias (Gomes Pereira & Wicherson, 1999), etc.

O LiDAR é um sensor activo que funciona como um radar ordinário, só que emite pulsos de luz em vez de ondas de rádio (Baltsavias, 1999; Wehr & Lohr, 1999; Pereira,

---

[2] É a ciência que trata dos métodos de observação da Terra por sensores instalados em satélites artificiais (ou aviões).

[3] Modelo numérico que representa a elevação do terreno conjuntamente com os objectos que se encontram sobre a superfície.

[4] Modelo numérico que representa a elevação do terreno "descoberto", isto é, sem os objectos que se encontram sobre a superfície.

2005). O sensor está constituído por um transmissor laser que gera pulsos ópticos, que ao entrar em contacto com um objecto são reflectidos e recolhidos através de um receptor óptico-electrónico. A velocidade da luz é conhecida, e um contador de alta velocidade mede o tempo desde que o pulso é emitido até ao momento que é devolvido, e finalmente este é convertido numa distância.

Ao ser montado numa aeronave (avião ou helicóptero), para poder referenciar correctamente o ponto que se mediu no terreno, utiliza-se a combinação de duas tecnologias diferentes: a) um Sistema de Navegação Inercial (INS) e uma interface de Unidade de Medição Inercial (IMU), que permitem medir a orientação exacta do sensor e as mudanças de atitude do avião, podendo em cada momento calcular as coordenadas exactas do ponto que está a ser medido; b) GPS diferencial (DGPS) para poder medir a posição exacta do sensor e um GPS cinemático aerotransportado que segue o rasto de pelo menos 4 satélites de navegação e regista a posição espacial exacta do avião.

Em áreas de vegetação (um dos primeiros ambientes a ser investigado com esta tecnologia), o raio choca em primeiro lugar com a copa da árvore. Nesse momento parte do raio é reflectivo e regressa à aeronave, no entanto ao tratar-se de uma superfície não sólida, há outra parte do raio que atravessa a vegetação até chegar ao solo e retorna à aeronave. Os sistemas mais comuns, normalmente guardam o primeiro e último eco (ou retorno). As propriedades do LiDAR foram já estudadas extensivamente (Kraus e Pfeifer, 1998; Gomes Pereira & Wicherson, 1999; Maas, 2002, Crombaghs *et al*., 2002; Ahokas *et al*., 2003), e ficou demonstrado que a elevação do terreno pode ser adquirida com uma precisão de 15 cm (Informação mais detalhada sobre esta tecnologia consultar Anexo I).

Os recentes avanços tecnológicos facilitaram o desenvolvimento de sistemas de laser pulsado em aeronaves que conseguem em função do tempo, a digitalização e gravação completa da forma de onda, isto é, de todo o sinal laser reflectido pelo objecto (normalmente designados por sistemas *full waveform*). Este sucesso abre novas potencialidades no uso desta técnica, visto que é possível obter um numero ilimitado de retornos (Hug et al., 2004; Wagner et al., 2004; Jutzi & Stilla, 2006; Wagner et al., 2006), conseguindo assim gerar densidades "verticais" de pontos mais elevadas que os sistemas convencionais (Persson et al., 2006; Reitberger e tal., 2007). Além do mais,

para cada retorno, é possível extrair informação adicional, tais como, a amplitude, a largura

do pulso e tempo de GPS do mesmo (Ducic et al., 2006; Ullrich e tal., 2007; Mandlburger et al., 2008).

O resultado de um voo LiDAR é uma densa nuvem irregular 3-D de pontos (X, Y, Z) que descreve a forma da superfície num determinado momento, ou seja, os pontos que definem o solo e os que definem os objectos aparecem sobrepostos. Nesse sentido, umas das primeiras operações necessárias independentemente da natureza do estudo é a filtragem e/ou classificação dos dados (geralmente designadas por operações de pós-processamento).

Actualmente, diversos algoritmos de filtragem foram já desenvolvidos, onde a maior parte deles utiliza operações de vizinhança local sobre os pontos para classificar os dados, em pelo menos dois níveis: terreno e os que não pertencem ao terreno (Sithole, 2005).

No entanto, a maior parte dos estudos comparativos de performance entre diferentes aproximações (Sithole & Vosselman, 2004) centram-se na eficácia da filtragem propriamente dita não contemplando a velocidade de processamento que consideramos fundamental devido à quantidade de dados que são necessários analisar e processar, visto que esta tecnologia foi concebida para realizar rápidos levantamentos à escala regional e nacional.

Numa malha regular de pontos (pixéis), como é o caso dos métodos que realizam uma interpolação prévia ou trabalham com outro tipo de dados (imagens aéreas ou de satélite), é possível estabelecer de imediato relações de vizinhança local dado que é estabelecido um tamanho fixo de pixel (tipo matriz) que garante um valor de elevação de forma regular, “amenizando” o problema até determinada dimensão de dados.

Se tivermos em conta que a maior parte dos algoritmos que foram desenvolvidos são iterativos e aplicados directamente sobre a nuvem irregular de pontos (de forma a não desvirtuar a informação), temos um problema acrescido dado que, no inicio do processamento não são conhecidas quaisquer relações entre os pontos que por exemplo, permitam conhecer o tipo de proximidade espacial.

Sendo assim, é necessária uma investigação sobre técnicas que permitem optimizar o tempo de processamento deste tipo de algoritmos para que estes possam responder às exigências de operacionalidade. Este trabalho será realizado, no âmbito do projecto de investigação (PTDC/AGR-CFL/72380/2006) intitulado "*Forest and Fuel Variables Estimation and Digital Terrain Modelling with Airborne Laser Scanning and High Resolution Multi-Spectral Images*", que pretende desenvolver (na forma de uma aplicação informática), implementar e validar uma metodologia que pela combinação de dados obtidos por LiDAR com aqueles derivados de imagens multi-espectrais possibilite a modelação do terreno, a inventariação rápida e fiável do seu coberto florestal e o cálculo de variáveis de combustível, necessários para uma adequada gestão florestal e determinação do risco de incêndio, prevenção e controlo.

Ainda que não seja o objectivo principal do projecto resolver de imediato este problema associado ao tempo de processamento, existe já uma preocupação emergente de delinear um novo caminho de investigação que permita num futuro próximo a sua integração com as soluções que vão ser alcançadas.

Nesse sentido, como ponto de partida, e que deu origem à presente dissertação, foi proposto realizar uma análise de custos de processamento (ganhos / perdas) de alguns recursos bastante associados a questões de performance, como é o caso da memória RAM e disco. A primeira conhecida como um recurso de altos desempenhos em acessos de leitura / escrita (nanossegundos), no entanto, limitado em termos de capacidade de armazenamento, e o segundo consideravelmente menos limitado em capacidade de armazenamento, mas muito mais lento no que diz respeito a tempos de acesso de leitura / escrita (milissegundos).

Além disso no caso de se utilizar métodos baseados em acessos ao disco pretende-se, comparar qual o ganho de optimização utilizando duas estruturas de dados espaciais adequadas ao tipo de geometria e dimensão espacial dos dados LiDAR.

Este estudo foi focalizado sobre um método de filtragem baseado em morfologia matemática para extrair MDT[5], que dadas as suas características carece do mesmo problema de performance, de outras aproximações desenvolvidas.

1.2 Descrição do Problema

No presente momento, dependendo do sensor LiDAR utilizado e de alguns variáveis relacionadas com a calibração dos instrumentos no seu conjunto, o resultado de um voo LiDAR, poderá ser uma nuvem de pontos com densidades que andam entre 4 a 20 pontos / m^2. Nesse sentido, o cálculo de MDT, envolve trabalhar com elevadas quantidades de dados, consistindo em muitos casos milhões de pontos.

Na derivação de MDT, três grandes linhas de orientação conceptual foram desenvolvidas para separar os pontos que pertencem ao terreno dos que pertencem a objectos. Alguns autores utilizaram uma função de descriminação baseada numa aproximação inicial da superfície: Em Kraus & Pfeifer (1998) foi utilizada uma predição linear para gerar uma primeira superfície que através de uma função iterativa atribui pesos aos pontos baseando-se numa análise de resíduos. Os pontos com pesos altos têm maior probabilidade de pertencer à superfície terrestre final. En Pfeifer et al. (2001) este método foi integrado num processo hierárquico para melhorar os resultados em zonas urbanas. Outros métodos baseados na mesma orientação foram desenvolvidos (Schickler et al., 2001; Briese y Pfeifer, 2001; Lee y Youman, 2003Petzold et al., 1999; Lindenberger, 1993; Elmqvist e tal., 2001).

Noutras aproximações só foi necessária a identificação de alguns pontos do terreno para a partir dai construir a superfície: Axelsson (2000) utilizou um método baseado numa deificação progressiva de uma rede irregular de triângulos (TIN), onde a partir dos pontos mais baixos distribuídos sobre a malha, construi-se uma primeira superfície a partir de um modelo TIN. De iteração para iteração vão-se adicionando pontos aos triângulos através da comparação dos ângulos dos triângulos cujos pontos não pertencem à TIN com os que pertencem. Se um ponto tem um ângulo inferior a um

[5] É um dos produtos cartográficos que está bastante associado à tecnologia LiDAR, e que serve de base de trabalho para muitos outros projectos relacionados com a gestão e ordenamento do território.

limitado recalculado em casa iteração, é adicionado a superfície final. O processo termina quando não existam mais pontos por debaixo do limitador. Um processo semelhante foi desenvolvido por Sohn & Dowman (2002).

Finalmente, outros autores centraram-se na combinação de operações de morfologia matemática (Haralick y Shapiro, 1992) para comparar diferenças altimétricas (Weidner & Fostner, 1995; Killian et al, 1996; Hug & Wehr (1997; Lohman et al. 2000). Em Vosselman & Maas (2001) foi desenvolvido um filtro baseado no declive, utilizando operações morfológicas de erosão, onde o operador é geralmente um funil invertido que procura os dados que se situam debaixo dele. Outras variantes deste filtro foram também desenvolvidos (Sithole, 2001; Roggero, 2001).

Zhang at al. (2003) baseando-se nas aproximações de Killian et al. (1996) e Lohman et al. (2000), desenvolveram um filtro morfológico progressivo. Depois de interpolar os datos LiDAR a partir do mínimo vizinho mais próximo, inicia um processo iterativo de operações morfológicas de abertura (erosão seguida de uma dilatação) acompanhadas pelo aumento, linear ou exponencial, do tamanho do elemento estruturante (chamemos-lhe kernel). Além disso, em cada iteração é calculada a diferença de elevação com a iteração anterior, atendendo a um determinado limite que pode ser constante ou não, dependendo da evolução do tamanho do kernel (linear ou exponencial). Os pontos são finalmente classificados como solo ou objectos.

Gonçalves-Seco et al (2006), baseado na aproximação de Zhang et al (2003) desenvolveram um filtro morfológico adaptativo (FMA), para zonas rurais, onde o filtro foi aplicado directamente sobre a nuvem irregular de pontos e foi proposta outra evolução linear do tamanho do kernel e do cálculo do limite que faz classificação dos pontos. Além disso, utiliza uma função de descriminação baseada numa segmentação vertical que permite em zonas florestais identificar pontos pertencentes à vegetação baixa e alta. Este método posteriormente foi alvo de algumas melhoras (Gonçalves-Seco, 2007) (ver Figura 1).

Como se pode verificar, os vários algoritmos desenvolvidos até então maioritariamente necessitam de realizar iterativamente operações de vizinhança local (alguns até análises de co-variância (Kraus e Pfeifer, 1998)), entre um ou mais pontos, para estabelecer entre um ou mais pontos, relações de proximidade e diferenças de cota (Sithole & Vosselman,

2004), de forma a atribuir algum factor de descriminação que funcione como um classificador.

No caso dos métodos aplicados directamente à nuvem irregular de pontos, essas relações de proximidade só são possíveis através de um cálculo prévio das distâncias euclidianas entre todos os pontos. Ora, este processo dependendo da quantidade de dados poderá traduzir-se num elevado custo processamento em tempo.

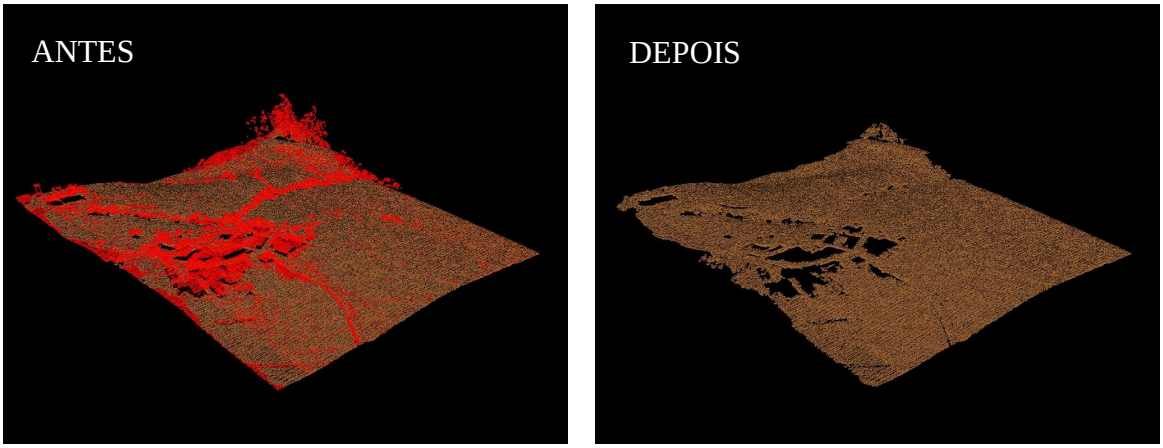

Figura 1. Resultado visual da aplicação do FMA.

Do ponto vista conceptual, os filtros morfológicos dependem de um elemento estruturante, isto é, de uma forma geométrica (que pode ser um quadrado ou uma circunferência) que percorre todos os dados e a partir de determinado tipo de operações morfológicas gera uma nova superfície que servirá para estabelecer relações de cota com a anterior, ora vejamos o princípio de funcionamento geral do FMA:

Para cada pontos ($P_{x,y.z}$) é aplicada uma função iterativa de classificação dos dados de terreno, onde a partir de um *kernel* 2D (circular e/ou quadrado) de determinado diâmetro (V_k) centrada em cada ponto é aplicada uma operação morfológica de erosão (Equação I.1), operação esta que se realiza a toda nuvem irregular de dados LIDAR.

$$e_p = \min\left(x_p, y_p\right) \in v_k(z_p) \qquad [I.1]$$

onde (x_p, y_p, z_p) são os vizinhos do ponto p situados a uma distancia menor ou igual que o tamanho do kernel (V_k) na iteração k. À superfície erodida é aplicada uma operação morfológica de dilatação (Equação I.2) com o mesmo tamanho do *kernel.*

$$d_p = \max\left(x_p, y_p\right) \in v_k(z_p) \qquad [I.2]$$

onde (x_p, y_p, z_p) são os vizinhos do ponto p situados a uma distancia menor ou igual que o tamanho do kernel (V_k) na iteração k. É calculada a diferença altimétrica entre a nuvem de pontos anterior e o resultado da operação de abertura (erosão seguida de uma dilatação). Os pontos são considerados objectos caso a diferença é maior que um determinado limitador, que é recalculado de forma iterativa em função do tamanho do *kernel* e de um declive (também recalculado). Em cada iteração, só é submetido a um processo de classificação os pontos que ainda não foram considerados objectos. O processo termina quando o tamanho do kernel é maior ou igual a um kernel máximo que deve ser atribuído a partir do objecto que tem a maior largura. Existe também um limitador máximo que deve coincidir com o objecto mais alto.

Ao ser um processo iterativo em que de iteração para iteração o raio de busca vai sendo maior, será necessário em cada iteração buscar para cada ponto os seus vizinhos que estão a uma distância igual ou inferior a um determinado raio.

A Figura 1 ilustra os resultados obtidos da aplicação do FMA sem qualquer tipo de optimização, em diferentes quantidades de pontos. Este processo foi realizado na memória RAM, isto é, os dados LiDAR são carregados directamente para a memória principal antes do processo de filtragem iniciar, não sendo realizado qualquer acesso ao disco duro.

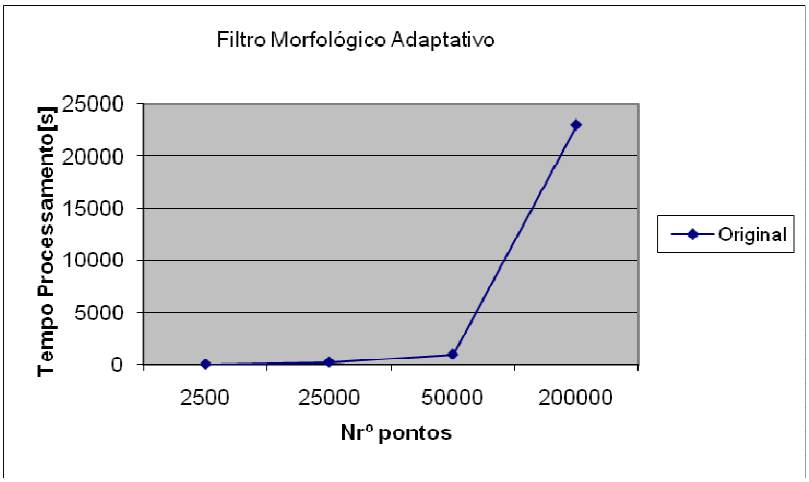

Figura 2: Tempo de processamento realizado com quatro iterações

Como podemos observar claramente, a partir de determinada quantidade de dados o tempo de processamento sobe exponencialmente com o aumento do número de pontos. Isto deve-se ao facto de que em cada iteração é necessário efectuar um recalculo das distâncias euclidianas para determinar quais os pontos vizinhos que se encontram dentro do kernel, esse aumento fazem com que esse processo demore cada vez mais de iteração para iteração com o aumento do raio de pesquisa. Sendo assim, o maior custo computacional é verificado nas operações de Abertura onde V é o tamanho do kernel do filtro e N é o número total de pontos. Então, para K iterações a complexidade do algoritmo é dado pela seguinte taxa de crescimento.

$$O(\sum_{k=1}^{k} V_k N) \qquad \text{[I.3]}$$

O processo de busca é um factor crítico na manipulação dos dados espaçais. Quando os dados não estão organizados espacialmente simples pesquisas como “procurar todos os pontos que estão a uma distância de 2 metros de um determinado ponto” requer a pesquisa sobre a base de dados inteira.

Nesse sentido, é necessário utilizar mecanismos espaciais que permitam organizar o espaço multidimensional e os objectos espaciais contidos nesse espaço de tal forma, que apenas uma parte do mesmo e um subconjunto dos objectos espaciais sejam considerados para responder a uma certa pesquisa espacial. Os métodos de indexação multidimensionais destacam-se por permitirem organizar este espaço, optimizando o desempenho na recuperação dos dados. Esta organização é crucial dado que cada dia o desenvolvimento deste tipo de tecnologias tem vindo a proporcionar a recolha de grandes quantidades de informação.

Além disso, dado que a capacidade de armazenamento da memória RAM é limitada, a análise do custo de acessos ao disco baseado em estruturas de dados espaciais optimizadas é fundamental para em futuras investigações tirar melhor partido deste dois recursos.

1.3 Objectivos

O principal objectivo desta dissertação consiste principalmente em avaliar o custo computacional avaliado em tempo de processamento de um filtro morfológico

manipulando dados em disco optimizados sobre dois métodos de indexação espacial e comparar a sua evolução em dimensão de dados com o mesmo método baseado em memória RAM. Nesse sentido, definiram-se os seguintes objectivos específicos:

- Análise e contextualização técnica sobre métodos de indexação espacial
- Optimização do FMA de forma a reduzir o número de pesquisas;
- Incorporação dos métodos de indexação com dados armazenados em disco no FMA optimizado;
- Avaliação do comportamento dos métodos de indexação.

1.4 Estrutura da Dissertação

A presente dissertação foi estruturada em cinco capítulos que reflectem genericamente as fases do trabalho desenvolvido de acordo com os objectivos propostos:

No Capitulo 1 foi realizada uma introdução ao trabalho, onde foram esgrimidas as motivações e formulação do problema que traçaram os objectivos propostos.

No Capítulo 2 é efectuada uma descrição um tanto técnica aliada a uma análise bibliográfica sobre as estruturas de indexação espacial para representação de pontos num espaço multidimensional, de forma a introduzir alguns conceitos que consideramos relevantes na compreensão deste tipo de estruturas. Finalmente são analisadas certas características, vantagens e desvantagens de cada uma das estruturas em relação ao tipo de geometria em estudo.

No Capítulo 3 são descritos todas as metodologias utilizadas para a consecução dos objectivos propostos.

No Capítulo 4 são apresentados e discutidos os resultados da metodologia apresenta.

Por fim no Capitulo 5 são apresentadas as principais conclusões tiradas do presente estudo e são apresentadas algumas perspectivas de desenvolvimento.

Capítulo 2

2. Métodos de Indexação Espacial

Devido a necessidade de processar grandes volumes de dados, e de dispor de mecanismos de indexação espacial que permitem organizar o espaço multidimensional de tal forma que no instante da pesquisa sejam considerados apenas uma parte deste espaço para dar resposta ao pedido, foi desenvolvido uma extensa investigação em torno das estruturas de indexações espaciais, a fim de agilizar as operações topológicas como é o caso do cálculo das distâncias e da vizinhança local. Entre os estudos realizados, destaca-se o trabalho de Gaede & Günter (1996) onde efectuaram uma extensiva revisão bibliográfica dos métodos de indexação multidimensionais. Posteriormente os trabalhos desenvolvidos centraram-se mais nas optimizações das soluções existente como vista a resolver problemas específicos como é o caso da sobreposição de rectângulos na R-Tree, surgindo assim estruturas como R*-Tree e R+-Tree.

Recentemente as investigações incidiram sobre métodos que permitem dividir o espaço multidimensional de pesquisa em blocos de tamanho fixo, armazenadas em páginas na memória ou em disco, mantendo uma descrição sobre o conteúdo de cada bloco (estruturados em geral de forma hierárquica), de uma forma que no instante de pesquisa, a informação possa ser acedida mais eficientemente. Uma revisão bibliográfica bastante extensiva pode ser encontrada em Samet. H (2007). Além desses estudos pode ser encontrado em Beomseok et.; al. (2007) uma comparação entre vários métodos de indexação espacial representantes da duas principais famílias.

Na secção 2.1 é realizada uma breve revisão sobre dados espaciais, abordando os tipos de dados espaciais existentes, as aplicações de dados e as pesquisas mais comuns realizadas sobre esses dados.

Na secção 2.2 é efectuada uma revisão bibliográfica sobre as estruturas de indexação espaciais propostas na literatura científica que podem ser aplicados aos dados LIDAR, analisando os pontos fortes e fracos de cada método.

Na secção 2.3 é efectuada uma revisão tecnológica sobre os Sistemas de Gestão de Bases de Dados com componente espacial.

2.1 Dados Espaciais

Dados espaciais podem ser aplicados em vários domínios, incluindo Sistemas de Informação geográfica (GIS), robótica, computação gráfica, realidade virtual, assim como em disciplinas, como análise de elementos finitos, modelagem de sólidos, design de computadores e fabrico, estatísticas, e design de circuitos lógicos.

2.1.1 Tipos de Dados

Dados espaciais consistem em objectos compostos por pontos, linhas, regiões, rectângulos ou volumes, que podem ser usados para representar a geometria de entidades geográficas como cidades, rios, estradas, montanhas, etc. Muitas vezes, também são usados atributos não-espaciais, normalmente designados por temáticos, para representar informações como elevação, alturas, cidade ou nomes para essas entidades. Um objecto espacial ocupa uma certa região do espaço, denominado de extensão espacial, que é caracterizado pela sua localização e por um limite.

Cada vez mais, aplica-se os dados espaciais para gestão ambiental, planeamento urbano e gestão de recursos. A Figura 6 ilustra alguns tipos de dados espaciais.

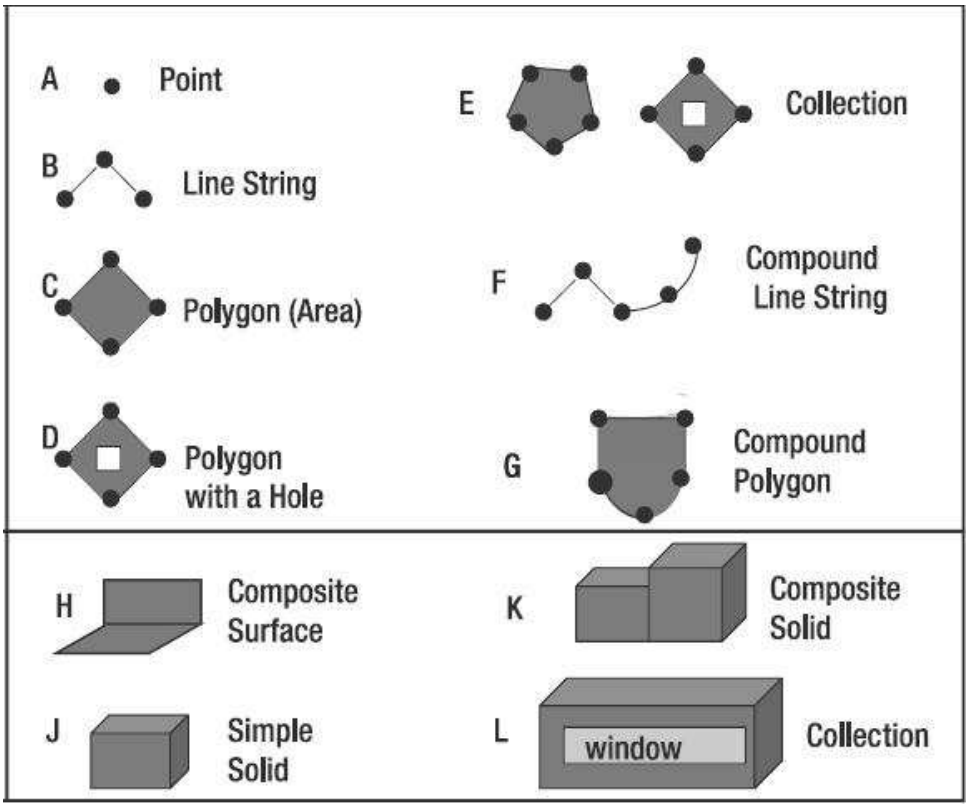

Figura 3: Tipos de dados Espaciais, fonte [RAE07]

2.1.1.1 Ponto

Um ponto contém uma extensão espacial caracterizado completamente pela sua localização, consequentemente não ocupa espaço não tem área nem volume associado. Conforme foi referenciado no Capitulo 1, dados LIDAR consistem em dados do tipo ponto. Exemplo: Objectos A e E na Figura 3.

2.1.1.2 Região

Dados do tipo região contem uma extensão espacial e um delimitador. A localização pode ser vista como uma posição fixa da região, por exemplo um centróide. Em duas dimensões o delimitador pode ser vista como uma linha (para regiões finitas, circuitos fechados), e em três dimensões como uma superfície. Exemplo: Objectos J, H na Figura 3.

2.1.1.3 Linhas

Linhas permitem conectar múltiplos pontos (ou vértices). Em geral é usada para representar estradas, ligações ferroviárias, redes eléctricas e entre outros. Uma linha é usada para conectar dois ou mais pontos da seguinte forma:

- **Rectas*:*** Quando existe uma linha recta ou uma linha onde não existe ambiguidades. Exemplo: Objecto B na Figura 3.
- **Arcos Circulares*:*** Representado por arcos circulares.
- **Combinação de rectas e arcos:** Normalmente é representada por uma linha composta (curva). Objecto F na Figura 3 é um exemplo duma curva.

2.1.2 Pesquisas

A manipulação dos dados espaciais, é um requisito fundamental que se aplica em vários domínios, incluindo aplicações ordenamento de território, aplicações de utilidade pública como transportes, redes de electricidade, águas e saneamento, ou serviços baseados na localização recorrendo á tecnologias como comunicações móveis e o GPS. É comum serem realizadas também operações topológicas como operações de vizinhança local e cálculos de distância entre objectos. Devido á importância dessas operações neste estudo, é feito uma breve caracterização sobre das mesmas.

2.1.2.1 Operações de vizinhança local

Operações de vizinhança local são utilizados para retornar conjunto de pontos que são vizinhos de um dado ponto de interesse *p*. No entanto existem deferentes ideias sobre o que constitui esta operação. Alternativas incluem todos os pontos que estão dentro de um raio de pesquisa, centrado em *p* Isaaks, E. H.; Srisvatra(1998).

Como não existe um consenso sobre a definição de vizinhança local, assumimos esta operação como dado um ponto *p*, encontrar todos os pontos *x* num conjunto *P*, de dados tal que a distância de $d(x, p) \leq r$, onde r é o raio de um círculo centrado em *p*. Este tipo de operações pode ainda ser aplicado a outros tipo de entidades mais complexas representadas, por exemplo, por regiões.

2.1.2.2 Operações por intervalo

Normalmente chamado de *range query* ou *window query*, especificando um rectângulo de arestas paralelas aos eixos do sistema de coordenadas, retornando todos os objectos que o interceptam. Este tipo de operação é bastante comum, nomeadamente para seleccionar os objectos que se encontram na área de visualização do ecrã do utilizador, para aplicação de filtros restringindo o volume de dados a serem analisados para a elaboração de operações complexas. A Figura 4 ilustra um exemplo deste tipo de operação.

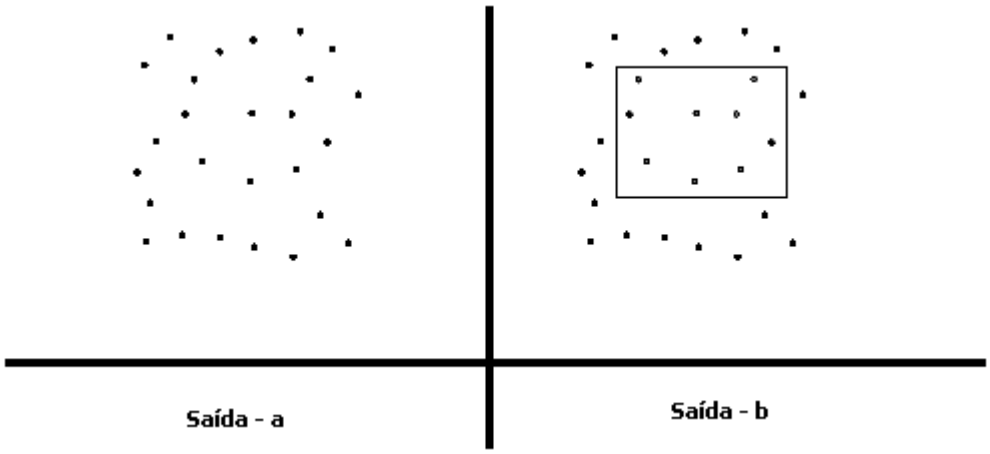

Figura 4: Range Search

Este tipo de operação é bastante eficiente, dado que evita percorrer toda a amostra de dados, mas é preciso que a informação espacial esteja organizada espacialmente. Indexação Espacial

Métodos de indexação espacial são técnicas para indexar e recuperar informação em conjunto de dados vectoriais. Segundo (Guttman, 1984), dados espaciais que envolvem áreas em espaços multidimensionais não são bem representados por pontos no espaço. Dados de aplicações CAD, e aplicações SIG possuem essa característica. Uma operação comum desse tipo de aplicação seria, alguns cálculos de vizinhança local como encontrar todos os objectos que estão incluídos dentro de uma determinada área, ou calcular a distância de um ponto a um conjunto de pontos. Para realizar essas operações é necessário criar relações entre os dados de maneira de maneira que nos cálculos de operações topológicas, algumas partes do espaço e um subconjunto dos objectos espaciais armazenadas sejam consideradas para dar resposta à uma pesquisa espacial.

Um método de indexação baseado na localização do objecto é então necessário para que esse tipo de operação seja realizado de maneira eficiente. Com o objectivo de indexar pontos da forma ($x1, x2, \ldots; x_n$) estruturas clássicas de dados unidimensionais como *hash tables,* ou *B-Tree* (R. H. Günting. & H. P. Kriegel, 1980) não funcionam para dados multidimensionais, dado que essas estruturas trabalham com comparações de igualdade entre variáveis enquanto aplicações espaciais necessitam de realizar comparações entre intervalo. Segundo (Sellis, 1987), métodos de indexação espacial são estruturas de dados que permitem o acesso a dados com mais de uma dimensão de maneira eficiente. Diversas estruturas de dados foram desenvolvidas com o propósito para representar pontos num espaço geográfico (2D ou 3D). Entre esses métodos existem essencialmente duas grandes famílias:

1. Estruturas de dados baseados na decomposição do espaço, conhecidos também como *Point Structures* (QuadTree (Finkey & J. L. Bentley 74), K-d Tree (J. L. Bentley 75) Grid-File (Donald, 2000) multidimensional B-Tree (Harry Leslie et., al., 71), e as suas variantes como MX CIF QuadTree (Kedem, 1982), PR QuadTree (Bentley, 74), BD-Tree (Abe et. al., 1993).
2. Estruturas de dados baseadas na distribuição espacial dos objectos (R-Tree (Guttman, 1984), e as duas variantes como R^*-Tree (Beckman, 1990), R^+-Tree (Beckman et; al, 1993).

Recentemente foi proposta uma nova família de estruturas de indexação espacial, consistindo numa junção das duas apresentadas em cima. Entre essas estruturas destacam-se os seguintes (SR-Tree (Katayama & S. Satoh), SS-Tree (White & Jain, 1996) e X-Tree (Berchtold & et. al, 1996)).

O estudo realizado nesta dissertação incidiu sobre métodos multidimensionais de indexação adequados indexar dados LIDAR. No entanto alguns métodos podem ser generalizados a outros tipos de dados.

2.1.3 Métodos baseados na decomposição recursiva do espaço

Esta família de indexação multidimensional tem sido amplamente usada para indexar dados multidimensionais do tipo ponto. Usualmente são utilizados para pesquisas de vizinhança local ou *range search*, sendo este o ponto forte que caracteriza esta família (Samet, 1984).

Entre os métodos destacados a QuadTree é o principal representante desta classe, sendo por isso bastante utilizada na investigação.

Para mais informação sobre métodos desta família e adequados a operações de vizinhança local podem ser encontrados em (Samet, 2007)

2.1.3.1 QuadTree

Segundo (Samet, 2007)**,** o termo QuadTree é usado para descrever uma classe de estruturas de dados hierárquica cuja propriedade comum é o princípio da decomposição recursiva do espaço. A QuadTree podem ser usada para representar um leque bastante alargado de dados, como pontos, áreas, curvas, superfícies e volumes. A decomposição pode ser regular (divisão em partes iguais a cada nível) ou determinada por características dos dados.

Em Sistemas de Informação Geográfica (SIG), utilizando esta estrutura de dados para realizar a indexação espacial, é considerado que cada nó corresponde à uma região quadrada do espaço. Esta região é dividida, recursivamente em quatro quadrantes, criando mais um nível na estrutura e assim sucessivamente até existir zeros ou um objecto dentro dos quadrantes resultantes das subdivisões.

Como exemplo temos a *QuadTree* do tipo região para a decomposição de regiões e *QuadTree* do tipo ponto para a representação de pontos num espaço multidimensional.

2.1.3.2 QuadTree do Tipo Região

Este tipo de *QuadTree* é usado para representar regiões bidimensionais. Uma vez definidos os limites da região com base na sua borda, esta estrutura baseia-se na subdivisão sucessiva do espaço em quatro quadrantes de mesmo tamanho. O critério usado para decomposição do espaço consiste em efectuar uma nova subdivisão enquanto este não for homogéneo, ou seja, possuir apenas elementos iguais.

2.1.3.3 QuadTree do Tipo Ponto

O QuadTree do tipo ponto é usado para representar um conjunto de pontos num espaço bidimensional. A principal diferença em relação à QuadTree do tipo região centra-se na decomposição do espaço. A decomposição ocorre em função da entrada, permitindo a formação de quadrantes rectangulares. Esta estrutura é considerada como uma adaptação da árvore de pesquisa binária (Donald, 97), onde cada ponto, com coordenadas Xp e Yp, divide o seu quadrante em quatro novos quadrantes da seguinte forma:

- Quadrante NO: X < Xp e Y > Yp
- Quadrante NE: X > Xp e Y > Yp
- Quadrante SO: X < Xp e Y <Yp
- Quadrante SE: X > Xp e Y < Yp

A inserção de um novo ponto consiste em percorrer a árvore a partir da raiz até alcançar uma folha, a qual corresponde ao quadrante sem subdivisão que contém ponto, onde este será inserido. A descida na árvore é efectuada comparando, segundo as regras acima, as coordenadas do novo ponto com as do ponto do ramo que vai sendo percorrido. Cada ponto adicionado à estrutura sempre leva a criação de quatro novos quadrantes.

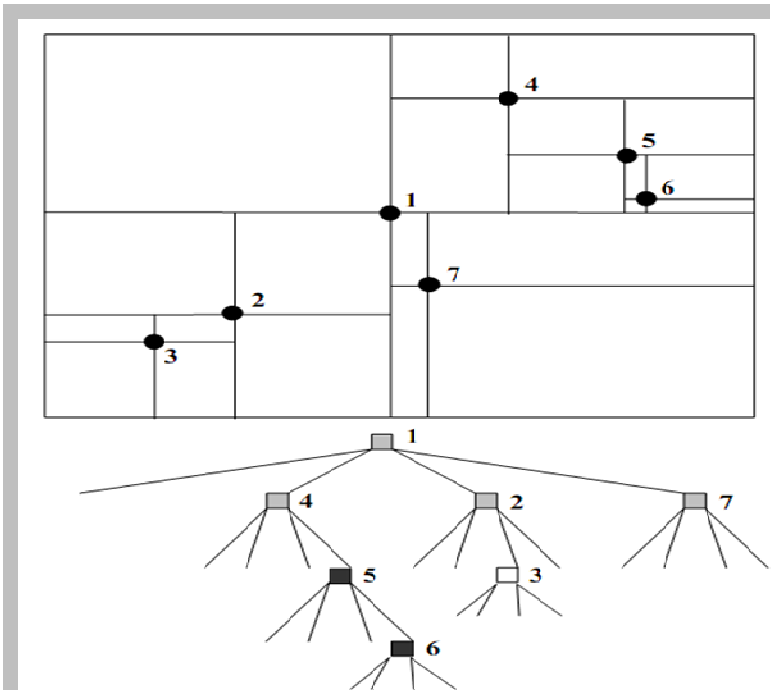

Figura 5: QuadTree do tipo ponto e a respectiva árvore representação

O formato da árvore é altamente dependente da ordem de inserção dos pontos, assim como ocorre na árvore de pesquisa binária.

A QuadTree apresentada na Figura 5, é um exemplo básico deste método. Várias metodologias foram propostas com o objectivo de resolver problemas específicos. A seguir descreve-se a MX- CIF QuadTree, uma das variantes da QuadTree.

2.1.4 MX-CIF QuadTree

A MX-CIF (*Caltech Intermediate Form*) *QuadTree*, pertencente á família das CIF QuadTree, foi criada independentemente por (Kedem, 1982) e (Abel, 1983).

Associa cada rectângulo com um nó da QuadTree correspondendo ao menor bloco que o contém inteiramente. Desta forma, podem existir rectângulos associados a nós internos. O processo de decomposição pára quando um bloco não contém inteiramente nenhum rectângulo. Um critério alternativo consiste em estabelecer um tamanho mínimo, de forma que o bloco só poderá ser subdivido caso o tamanho seja superior ao do limite mínimo. Kedem (Kedem, 1984) sugere que o limite seja igual ao tamanho esperado dos rectângulos.

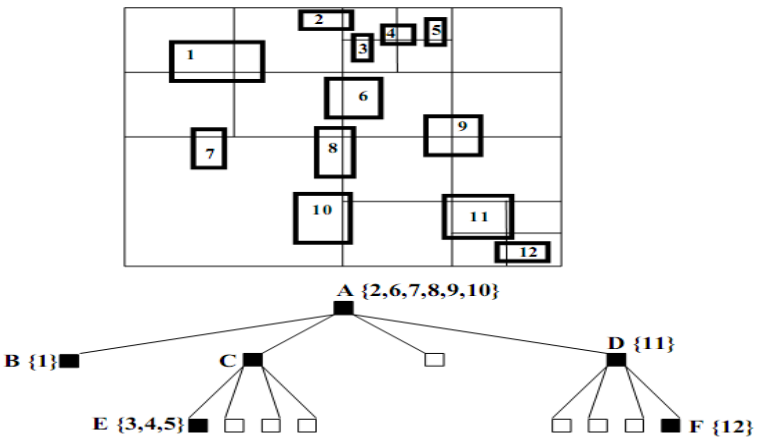

Figura 6: MX-CIF QuadTree a árvore representativa

A Figura 6 apresenta um exemplo da MX-CIF QuadTree. Cada rectângulo é associado apenas a um nó, e não é considerado membro dos filhos destes. Como exemplo o rectângulo 11 possui intersecção com os nós D e F porém só esta associado ao nó D. Por outro lado, mais de um rectângulo pode ser associado a um nó, existindo diversas formas de organizá-los. A MX-CIF QuadTree associa a cada rectângulo um nó da árvore. Outra abordagem possível é a representação através de múltiplos blocos ou nós.
A MX-CIF QuadTree expandida procura representar cada rectângulo de uma forma mais compacta no diz respeito à área. Proposta por (ABEL, 1985), esta representação segue os mesmos passos de uma MX-CIF QuadTree diferenciando-se quando o menor bloco que contém o rectângulo inteiramente é encontrado. Neste ponto um número prefixado de subdivisões é efectuado no bloco.

2.1.5 K-d Tree

É uma árvore de pesquisa binária de *k* dimensões que se caracteriza por testar, em cada nó percorrido, um valor chave. Esse valor chave está associado a um discriminador. O discriminador indica qual dimensão da k-d Tree foi utilizada para determinar o valor chave que divide os nós restantes em duas sub-árvore. Os nós com os valores dessas dimensão maiores que o valor chave são armazenados do lado direito e os restantes do lado esquerdo do nó percorrido. A kd-Tree é eficiente para consultas que envolvam cálculos de índices e proximidades. As consultas de proximidade geralmente são feitas com base em um ponto central e um raio de busca. O resultado é um conjunto de pontos contidos dentro dessa região circular (área de *buffer*).

Segundo [Samet06] existe diversas variações para a *kd-Tree* do tipo ponto. Essas estruturas diferenciam na maneira como o plano é feita a divisão, bem como no próprio armazenamento do dado espacial nos nós da árvore. Na *kd-Tree*, do tipo ponto o valor

do discriminador é o valor da uma das k-dimensões de cada nó da árvore. A escolha da dimensão que será utilizada como discriminador é realizada de forma alternada. Por exemplo, para uma kd-Tree de duas dimensões, denotada por *x* e *y*, o valor do discriminador do primeiro nível é o valor da dimensão *x* e o próximo discriminador o valor de *y*. No próximo nível, o discriminador volta a ser o *x* e assim sucessivamente.

Cada nó de uma kd-Tree contém no mínimo k + 4 campos. No caso de uma *kd-Tree* para pontos no plano, cada nó possui um total de seis campos. Desses campos, dois são utilizados para armazenar as coordenadas *x* e *y* do ponto; os outros dois são os apontadores para a sub-árvore da esquerda e para a sub-árvore da direita; um campo é o identificador do nó, e o último campo é utilizado para armazenar as coordenadas x e y do ponto; os outros dois são os apontadores para a sub-árvore da esquerda e para a sub-árvore da direita; um campo é o identificador do nó, e o último é utilizado como discriminador deste nível.

O processo de inserção de pontos numa kd-Tree é semelhante ao processo de inserção numa árvore binária. Para cada novo ponto a ser inserido na árvore, percorre-se a árvore verificando em cada ponto a posição em que este se encontra inserido. Essa verificação da posição ocorre a partir da comparação do valor de uma das coordenadas do novo ponto com uma das coordenadas do nó da árvore.

A coordenada escolhida para a comparação é indicada pelo discriminador do nó da árvore que esta sendo percorrido. Se o valor da coordenada do novo ponto for menor ou igual à coordenada do nó da árvore, então percorre-se a sub-árvore à esquerda do nó da árvore. Caso contrário percorre-se a sub-árvore direita. Caso o apontador para a sub-árvore a ser percorrida tiver o valor nulo, então o novo ponto é inserido nesse local e o apontador do nó da árvore é actualizado, passando a apontar para o novo ponto.

2.1.6 Métodos baseados na distribuição espacial dos objectos

Existem outros tipos de dados tais como: triângulos, quadrados, rectângulos que não podem ser indexados como pontos isolados, dado que são formados por mais de um ponto. Nesse caso não é difícil perceber que os métodos baseados na decomposição recursiva do espaço não seriam mais adequados. Dados com estas características podem ser indexadas utilizando métodos baseados na decomposição de objectos descritos a seguir.

2.1.6.1 R-Tree

A R-Tree é uma estrutura de dados baseada numa optimização heurística, derivada da estrutura B-Tree. As principais diferenças estão na natureza das chaves: valores numéricos ou alfanuméricos simples, no caso da B-Tree, e pontos extremos de rectângulos no caso da R-Tree (Gutman, 1984).

É aplicado a métodos de acesso espacial, ou seja é usado para indexar dados multidimensionais, como por exemplo (X, Y, Z), que correspondem as coordenadas de dados geográficos como pontos. Esta estrutura procura organizar não exactamente o contorno ou a forma gráfica do objecto, mas sim o menor rectângulo envolvente (MBR). Este rectângulo é formado a partir de observação dos limites geométricos mínimo e máximo do contorno do objecto, e é expresso pelas coordenadas dos seus pontos inferior esquerdo e superior direito.

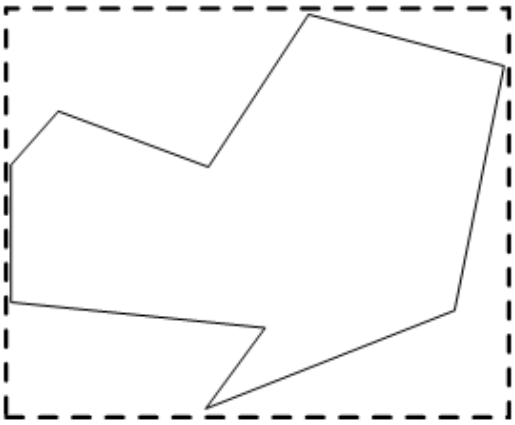

Figura 7: Objecto poligonal e o respectivo MBR

Cada nó corresponde uma página de disco, e a estrutura utiliza heurísticas durante a construção da árvore, de maneira que, numa consulta, o número de nós visitado e o número de operações de entrada e saída seja minimizado.

Os dados espaciais são representados como uma colecção de tuplos. Cada tuplo possui um identificador único que pode ser usado para recuperar a mesma. Cada nó tem entradas da forma de dados n-dimensional representando pelo tuplo indicado. Nos nós que não pertencem as folhas as entradas são rectângulos da forma (I, *apontador_filho*), onde I é o rectângulo mais pequeno que cobre todos os rectângulos dó nó filho. Outro tipo de nó que existe, são os ramos, estes normalmente contém vários pares (*cld_apontador*, *rect*) onde cl_ *apontador* é um apontador para um dos seus filhos e o *rect* é o rectângulo mais pequeno que se consegue englobar todos os rectângulos que pertencem ao seu filho.

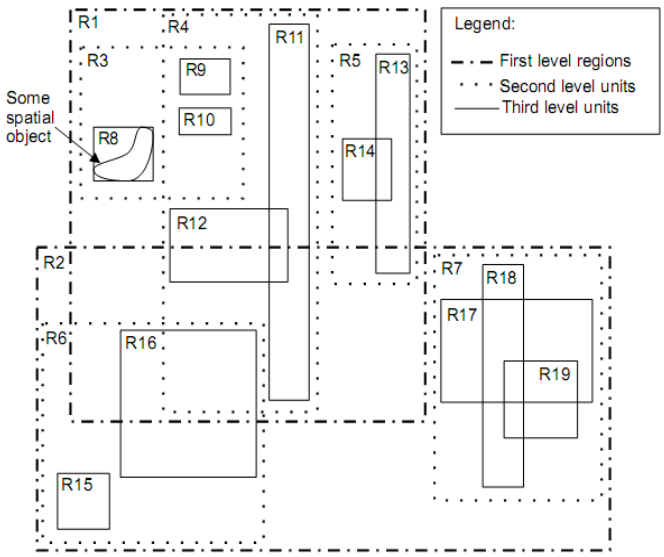

Figura 8: MBR representada por uma estrutura R-Tree

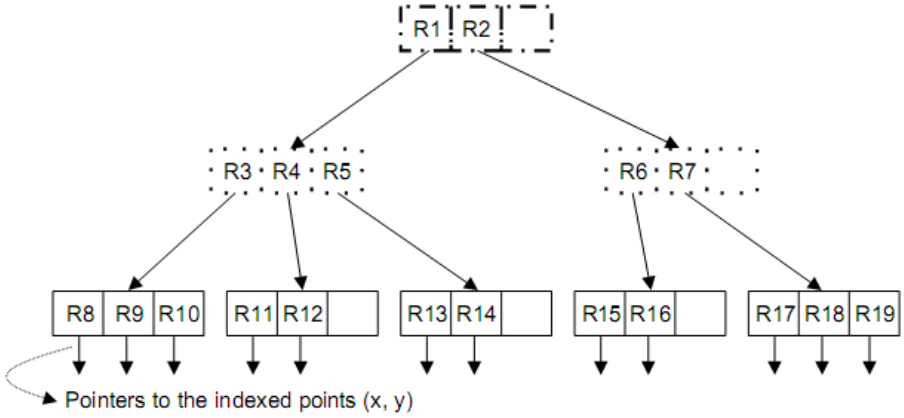

Figura 9: Representação esquemática dos dados armazenados na R-Tree

Cada nó desta estrutura tem de preencher um determinado número de entradas. Assim pode-se admitir que sendo *m* o número de nós e *N* o número máximo de nós, a relação entre eles é (*ni* <= *mi* <= *N*). Também existe uma relação entre *n* e N que se estabelece da seguinte forma: ni = N/2, em que a performance é maximizada quando n é exactamente igual a *N/2*. A altura de uma árvore é log n *M – 1* em que M é o número de tuplos. Em termos de ocupação o número máximo de nós que existe é M/n + M/n^2 + … + 1e o pior caso de utilização de nós é n/N.

A inserção de um nó começa-se pela raiz e vai-se comparando o (*Maximum Bound Rectangle*) *MBR* do objecto que vai sofrer a operação com os que existem dentro dos nós e escolhe-se aquele cujo rectângulo contém essa área, se nenhum deles contiver totalmente essa região, vai ser escolhido aquele que vai ter o menor aumento da sua área

para poder ficar com o objecto, repetindo este procedimento para cada nível até chegar as folhas, onde existem, 3 hipóteses: inserir directamente a nova referência do objecto, alargar o rectângulo do nível superior para encaixar este novo objecto ou fazer uma divisão (*split*) para se arranjar mais espaço, pois a folha encontra-se cheia. Quando é necessário fazer *split*, existem 3 algoritmos que permitem minimizar as áreas cobertas pelos rectângulos: um linear, um exponencial e quadrático. A diferença entre esses algoritmos reside no número de combinações feitas com os MBRs, antes da divisão para escolher quais são as melhores combinações.

2.1.6.2 R*-Tree

A R*-Tree é uma optimização da R-Tree. Pode representar dados do tipo ponto e região ao mesmo tempo. A grande diferença em relação a R-Tree reside na heurística de optimização. Enquanto a R-Tree só usa a área como parâmetro para inserir um novo elemento a R*-Tree usa a área, o perímetro e a área sobreposta dos MBRs. Outra diferença é que para não ser necessário criar novos nós, esta estrutura tenta maximizar o espaço de cada um deles [BKS90].

A minimização da sobreposição e *overflow* é crucial para o bom desempenho da R-Tree. A R*-Tree faz tentativas para reduzir os dois problemas, utilizando a combinação de um algoritmo do nó visitado e o conceito de forçar a reinserção no nó *overflow*.

Na inserção, a alteração dá-se ao nível da condição que é executada no nó que fica imediatamente antes da folha. Normalmente escolhe-se o ramo que implicava o menor aumento da área total do MBR, agora vai-se escolher o que implica o menor aumento da área que fica sobreposta. A maneira de fazer o *split* também foi alterada. Em vez de se dividir simplesmente os nós, encontrando uma *seed* e depois distribuir os objectos conforme a proximidade de cada um tem em relação à *seed* mais próxima tentando encontrar o conjunto de MBRs mais pequenos, vão-se ordenar os MBRs em relação a cada um dos eixos pelo valor mais baixo e depois pelo mais alto.

Um das características deste *split* é a reinserção forçada. No caso de uma inserção num nó que já esta cheia, em vez de haver um *split*, vai haver uma troca de elementos, em que o novo toma o lugar de um que já esteja inserido, e vai -se verificar se é possível colocar este numa outra folha com espaço livre, tentando assim evitar que se crie uma nova folha. Caso o elemento a ser deslocado não encontre uma folha que lhe dê acolhimento melhor do que aquela em que estava, então neste caso faz-se o *split*.

Conforme referido anteriormente, devido à quantidade de dados e a sua complexidade, o armazenamento dessas informações numa estrutura de dados seria impraticável, devido a dimensão de índice criado. Essa deterioração ocorre principalmente por dois problemas:

1. *Problema de fanout:*

   Sendo *fanout*, a quantidade de entradas que um nó é capaz de armazenar, estruturas famílias da R-Tree utilizam MBRs para agrupar dados que estão uma determinada sub-árvore, o que requer o armazenamento do MBR.

   Para dados bidimensionais, são necessário representar toas as coordenadas do MBR. Assumindo coordenadas reais, seriam necessários 4 números reais, o que gastaria pelo menos 16 bytes. Além disso, cada entrada também armazena um apontador para a sub-árvore ou para o seu objecto. O problema ocorre geralmente com o aumento da dimensão dos dados, que, devido ao custo da representação do MBR, torna o tamanho da entrada bastante grande. Com entradas grandes, poucas entradas são armazenadas em um nó, o que torna a árvore menos larga e mais alta, aumentando o desempenho da consulta.

2. *Problema de sobreposição*

   Em estruturas como a *R-Tree*, é possível ocorrer sobreposição de MBRs. O problema é que quando o grau de sobreposição dos MBRs de um mesmo nível é muito alto, durante uma consulta, é necessário que vários caminhos sejam percorridos. As heurísticas que mantêm essas estruturas não conseguem evitar o crescimento do grau de sobreposição entre os MBRs, quando a dimensão dos dados é muito grande.

A estrutura R^+-Tree reduz a sobreposição os MBR, repartindo os objectos sobrepostos e armazenando as partes em nós diferentes. Apesar de resolver este problema, este algoritmo cria a necessidade de propagar o particionamento estrutura abaixo. Um outro problema é que para propagar um objecto, várias sub-árvores poderiam ser acedidas por causa das partes que estão separados. Esse mesmo problema da divisão de um objecto afecta a remoção, pois muitas sub-árvores deverão ser percorridas para remover um único objecto.

Em relação à R^*-Tree, as principais vantagens são:

- Apresenta um novo algoritmo de inserção chamado *forced reinsertion* onde alguns objectos são inseridos novamente na árvore antes de particionar um nó;
- Ao contrário da *R-Tree,* que procura apenas minimizar o volume dos nós criados, este método tenta minimizar o perímetro e maximizar a ocupação dos novos nós.

Uma outra evolução da R-Tree é a X-Tree. Esta estrutura define um super-nó para resolver o problema de sobreposição de MBR. Caso o índice de sobreposição é alto, os

nós são concatenados em um super-nó de tamanho variável (geralmente um múltiplo do tamanho original). Essa abordagem é interessante só se for possível aceder o disco com páginas de tamanho variável. Dessa forma, há realmente uma diminuição no número de acesso a disco. Se isso não acontecer, como no caso dos SGBDs, o número de acessos aumenta e o desempenho pode ser próximo ao da R-Tree.

Muitas técnicas existentes são muitos sensíveis à dimensão dos dados. Isso significa que pontos próximos e algoritmos de busca por abrangência (*range search*) apresentam dependência exponencial em relação à dimensão dos dados [CHAZ94].

Informações mais detalhadas sobre métodos de indexação espaciais podem ser encontradas em H. Samet [SAM07]. Mesmo considerando o potencial dos métodos baseado na decomposição recursiva de objectos neste domínio, existem tipos de dados complexos que não podem possuem coordenadas e, consequentemente não podem ser indexados através desses métodos. Para resolver este problema, uma alternativa é utilizar os métodos de indexação métricos.

Alguns autores para evitarem trabalhar com métodos de indexação espaciais complexas, como triangulação de Delaunay, e estruturas de dados espaciais tradicionais como QuadTree e R-Tree propuseram novas metodologias bastantes simplificados utilizando apenas Vectores ordenados de dados. Entre esses estudos destaca-se o trabalho realizado por Thomas H. Meyer [THM06].

2.2 Base de Dados

Devido a necessidade de armazenar grande quantidade de informação e de as guardar localmente, é necessário escolher uma base de dados com extensão espacial que permite garantir não só que toda a informação é mantida de maneira segura, como também garantir um rápido acesso sobre os atributos espaciais dos dados armazenados, tanto para leitura como para escrita, disponibilizando funções auxiliares para o tratamento e análise dessas informações.

Nos últimos anos, além das bases de dados comerciais, vem surgindo um grande número de tecnologias baseadas em software livre permitindo trabalhar com vários tipos de dados espaciais, como linhas, pontos e polígonos. A possibilidade de usar a linguagem SQL, permite usar funções nativas para pesquisas espaciais.

A seguir são descritos alguns SGBDs que disponibilizam extensões espaciais.

2.2.1 PostgreSQL

PostgreSQL, foi inicialmente uma base de dados relacional e *open source*. Foi desenvolvido a partir do projecto Postgree em 1986 em Berkeley pela Universidade de Califórnia. È conhecida por ser fiável e multi-plataforma. Cumpre as normas ANSI-SQL 92/99, e é capaz de executar procedimentos que foram guardados, escritas em linguagens como Java, Perl, Pyton, Ruby, Tcl, C/C++/C# e PL/pgSQL, uma variação do PL/SQL da Oracle com quem é compatível. Disponibiliza ainda interfaces de comunicação com Java (JDBC), ODBC, Perl, Pyton, C / C++, . Net e entre outros [Post06].

Actualmente PostgreSQL é uma base de dados relacional-objecto e inclui um suporte ao desenvolvimento de extensões. Essa característica possibilitou o desenvolvimento de uma extensão, o PostGis, desenvolvido pela empresa Canadense Refractions Research Inc, sobre licença GNU GPL.

2.2.2 MySQL

MySQL é um projecto *open source* de uma base de dados relacional e cumpre as normas ANSI-SQL 92/99. Existem versões para vários sistemas operativos entre os quais *Windows, Linux* e *Solaris*. É possível comunicar com Java, C#, C/C++, no entanto desenvolvedores criaram driver que permite a ligação a outras linguagens nomeadamente: PHP, Perl, *Pyton,* Ruby e entre outros. Provavelmente de todas as bases de dados é aquela que possui menos suporte para operações espaciais e o facto de usar uma implementação da R-Tree com divisão quadrática vem atestar isso mesmo, já que este tipo de algoritmo de divisão foi o primeiro a ser desenvolvido e o seu uso não é muito comum.

2.2.3 Oracle

Actualmente o *Oracle* é o líder mundial em termos de base de dados. Existem várias versões, sendo a mais recente a 11g disponibiliza quatro edições diferentes: a *Standard Edition, Standard Edition One e a Enterprise Edition.*

O Oracle Spatial é a componente espacial do Oracle que vem integrado com a *Enterprise Edition 11g* e oferece as mais completas ferramentas para manipulações de dados geoespaciais. No Oracle Spatial é possível indexar dados de 2 a 4 dimensões usando objectos do tipo *SDO-GEOMETRY*.

Este objecto oferece através de tipos de dados geométricos, modelos de dados espaciais, *layers*, suporte para pesquisas, indexação espacial com R-Trees, QuadTrees e

geocoding. Existem outras opções de SGBDs que possuem extensões espaciais, como é o IBM DB2 Spatial Extender, e o Informix Spatial *Geodetic Datablade*. A tabela 1 apresenta um estudo comparativo sobre os SGBDs com componente espacial.

Recursos	***Oracle Spatial***	***PostgreSQL com tipos Geométricos***	***PostgreSQL com PostGIS***
Indexação espacial	R-Tree e QuadTree	R-Tree nativa ou R-Tree sobre GIST	R-Tree sobre GIST
Operadores topológicos	Matriz de Intersecções	Não	Matriz - Intersecções DE
Operadores de conjunto	Sim	Não	Sim
Operadores de *buffer region*	Sim	Não	Sim
Transformação entre sistemas de coordenadas	Sim	Não	Sim
Tabelas de metadados das colunas	Sim (conforme OGIS)	Não	Sim (conforme OGIS)

Tabela 1: Quadro comparativo entre SGBDs com extensão espacial

Na tabela 1 faz-se uma comparação das principais funcionalidades disponibilizadas pelos SGBDs com componente espacial.

2.3 Estratégia de Cache Espacial

Para evitar muitas operações de entrada e saída (I/O), a estratégia de *cache* espacial é essencial para aumentar a performance no processamento. O principal objectivo desta estratégia consiste em tirar partido ao máximo do efeito *cache*, de maneira que as informações possam ser acedidas mais rapidamente.

2.4 Tiling

Tiling consiste em dividir um espaço em áreas mais pequenas. Este é um método que pode ser usado para criar uma grelha que a sobrepõe ao terreno e que permite manter o registo de quais as secções que se encontrem carregadas. Outra característica desta técnica é a redução da quantidade de dados que é transferida, dado que apenas é transferido parte dos dados, aqueles que são realmente necessários. Esta quantidade é dependente do tamanho de cada *tiling level*, sendo necessário criar um equilíbrio entre a quantidade de dados transferida e o número de transferências através do tamanho do *tiling.*

Encontrar o tiling level correcto é crucial, dado que os quadrantes podem ser pequenos demais ou grandes demais [KT02]. No Oracle Spatial é preciso parametrizar o *tiling level* e o *numtilles*.

2.5 Sumário

Conforme citado na secção 2.2 existem essencialmente duas grandes famílias de estruturas de dados para a representação de pontos no espaço geográfico (2D ou 3D): estruturas baseadas na decomposição do espaço, conhecidos também como *Point Structures* (QuadTree, k-d Tree); estruturas de dados construídas sobre os objectos espaciais (neste domínio pontos), como são o caso da R-Tree e as suas variações; e outros propostos recentemente consistindo numa combinação das duas famílias, denominadas de estruturas híbridas (SR-Tree e X-Tree, SS-Tree).

Nesta fase do trabalho pretende-se essencialmente escolher dois métodos de indexação representantes das duas principais famílias.

Sendo a R-Tree a base de construção da sua família, é um dos métodos de indexação amplamente utilizado na investigação, e presente nos SGBDs tanto comerciais, como *open source,* foi um dos métodos escolhidos para indexar os dados LIDAR.

A segunda escolha recaiu sobre a *Point QuadTree*. Este método adequa perfeitamente aos dados LIDAR. Sendo a QuadTree um das principais representantes da sua família é largamente utilizada em projectos de investigação.

As duas estruturas escolhidas são indicados para indexar e optimizar pesquisas do tipo *range search ou nearest-neighbourd* (Samet, 2007).

Capítulo 3

3. Metodologia

Este estudo combina tarefas de implementação, experimentação e comparação. No presente capítulo será realizada uma descrição da metodologia que foi adoptada para concretizar os objectivos a que nos propusemos.

3.1 Dados utilizados

A área de estudo situa-se no norte da Galiza (Espanha) perto de uma paróquia chamada *Vilapena* (Espanha). É uma zona caracterizada por uma extensa plantação *Eucaliptus globulus* (em densidade por hectares, uma das mais altas da Europa), um relevo bastante diversificado e uma variação de altitude de 150 a 530 metros. Sobre a área de estudo foi definida uma área de 4 $km^{2,}$ através de um rectângulo de 1x4 km com o lado maior orientado na direcção do norte geográfico, segundo as seguintes coordenadas UTM com os extremos em (644800;4806600) e (645800; 4810600).

O varrimento laser, realizado em Novembro de 2004, foi efectuado com um sensro Opthec ALTM 2033, com uma frequência de repetição do laser pulsado de 33 kHz, com um ângulo de varrimento variando entre 0 a ±10º, e voado a uma altitude de 1500 m. A nuvem de pontos final tem uma densidade de ± 4 pontos / m^2.

3.2 Método de optimização do FMA

Com o objectivo de reduzir o número de operações de vizinhança local e analisar os ganhos que poderão significar (RAM e disco), foi desenvolvida uma optimização do FMA apresentado no Capitulo 1.

O FMA é um método iterativo que parte de um tamanho inicial do kernel e termina quando atinge o tamanho máximo do mesmo. Sendo esses diâmetros conhecidos visto que são parâmetros de entrada do próprio algoritmo (Gonçalves-Seco, 2007), a ideia foi partir do principio que conhecendo as distâncias do raio máximo (kernel / 2) seria possível determinar os pontos vizinhos para as restantes pesquisas sem voltar a calcular as distâncias. Nesse sentido foi implementado um método para processar as distâncias euclidianas entre os pontos para o diâmetro máximo:

```
public void ProcessaDistancias(double max_Kernel);
```

Este pré-processamento é realizado antes do processo de filtragem propriamente dito, onde para cada ponto é guardado o índice e as distancias dos pontos que estão dentro do raio máximo de busca no cálculo das distâncias. Essas distâncias são guardados numa estrutura de dados ordenados em memória, de maneira que em cada iteração com o novo *kernel* de pesquisa, não é preciso percorrer toda a estrutura de dados para devolver as distâncias que estão dentro desse raio de busca. Além disso, ao ter guardado o indice correspondente, o acesso à variável de elevação (necessário nas operações de *Abertura*), é imediata.

3.3 Criação da Base de dados espacial

Foi realizado um levantamento dos SGBDs com componente espacial instalada, tantos softwares comerciais como *open source*. Dos SGBDs analisadas na secção 2.3 a Oracle é a única que disponibiliza uma implementação dos dois métodos pretendidos, oferecendo funções e métodos bastantes optimizadas para a criação e manipulação dos dados espaciais. Ver Tabela1. A versão escolhida foi a *Enterprise Edition*, dado que oferece mais funcionalidades para a manipulação dos dados espaciais. Visto que a Universidade do Porto mantém um protocolo académico com a Oracle o problema da licença de uso não foi um obstáculo.

3.3.1 Criação do Índices Espaciais

Oracle Spatial consiste num conjunto de objectos: operadores, funções e procedimentos que usa esses objectos de diferentes tipos. Uma geometria é armazenada como um objecto, numa única coluna do tipo *SDO_GEOMETRY*. A criação dos índices espaciais

é feita usando comandos DDL básicos como (*CREATE, ALTER, DROP*) e DML (INSERT, UPDATE, DELETE).

Para criar um índice espacial e realizar pesquisas foi necessário executar as seguintes operações pela ordem indicada:

1. Criação da tabela espacial
2. Inserção dos dados na tabela
3. Actualização da *View USER_SDO_GEOM_METADATA*,
4. Criação do índice espacial
5. Realização das pesquisas

3.3.1.1 Criação da tabela Espacial

Para criar uma tabela espacial na base de dados Oracle foi necessário definir um conjunto de parametrizações. Como os dados LIDAR são dados do tipo ponto 2.5 D, foi preciso especificar a dimensão dos dados.

Para armazenar as coordenadas de cada ponto, criou-se uma coluna do tipo *GEOMETRY*, para a parti daí serem criados os índices espaciais. A Tabela 3 ilustra a estrutura da tabela espacial criada e no Algoritmo 1 o código SQL utilizado para criar a tabela.

Descrição	*Id.*	*Pontos*	*Intensidade*	*Fl Pulse*
Tipo	NUMBER	SDO_GEOMETRY	NUMBER	NUMBER

Tabela 2: Tabela com os tipos de dados criados para armazenar os pontos

```
CREATE TABLE fma (
  fma_id NUMBER PRIMARY KEY,
  pontos SDO_GEOMETRY,
  Int NUMBER
Pulse NUMBER);
```

SQL 1: Código SQL para criar uma tabela espacial

3.3.1.2 Inserção

Apesar do FMA utilizar elementos estruturantes de 2D, além das coordenadas X, Y, foi armazenada a Z, visto que é necessária nas operações morfológicas (em alternativa poderia ter sido armazenada fora da geometria).

```
INSERT INTO fma
VALUES
(
    Id,
    SDO_GEOMETRY (
    2001,
    20032629, --SRID: coordenada geodetic
    SDO_POINT_TYPE (x_cord, y_cord, Z_cord),
    NULL, NULL)
    Int NUMBER --Intensidade
    Pulse NUMBER -- Pulse
);
```

SQL 2: Código Sql para inserir dados na tabela espacial

3.3.1.3 Update Metadata View

Esta operação foi necessária antes dos índices serem criados. O objectivo foi fornecer as dimensões dos dados a analisar. Apenas foi necessário para a coluna do tipo geometria neste caso *pontos*.

```
INSERT INTO user_sdo_geom_metadata
(
      TABLE_NAME,
      COLUMN_NAME,
      DIMINFO,
      SRID
)
VALUES
(
      'FMATESTE',
      'pontos',
      SDO_DIM_ARRAY(
      SDO_DIM_ELEMENT('X',xmin,xmax, 0.005),
      SDO_DIM_ELEMENT (‘Y’, ymin, ymax, 0.005)
), 2032629 -- SRID
);
```

SQL 3: Código SQL para inserir os dados na tabela `user_sdo_geom_metadata`

3.3.1.4 Criação do índice espacial

A sintaxe para criação de índices espaciais é apresentada em baixo. O parâmetro *parameter_string pode* ser um conjunto de variáveis. Caso não seja especificado nenhum parâmetro SDO_LEVEL então é criado o índice R-Tree.

```
CREATE INDEX rtree_idx ON
FMA (pontos) – coluna do tipo Geometria
INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

SQL 4: Código SQL para criar índice R-Tree

Para a criação do índice QuadTree foi necessário especificar o número de subdivisões recursivas do espaço (*tiling level*) e o número de número máximo de objectos utilizado para indexar os rectângulos (*numtiles*). Para determinar o *tiling level* ideal mediante um volume de dados, foram necessários realizar um conjunto de testes (variando o número de *tiling* e o número de *tiles*), e contabilizar o respectivo tempo de processamento (ver Tabela 4).

		Tempo Processamento(s)			
Tiling	nrº pontos	tilles = 10000	tilles = 30000	tilles = 50000	tilles = 100000
	2500	0,4	0,38	0,37	0,37
	250000	15,86	17,74	16,09	18,8
2	2076834	169,55	172,08	169,14	168,32
	2500	0,37	0,36	0,37	0,38
	250000	15,71	15,9	16,17	16,18
4	2076834	168,33	167,89	167,2	168,61
	2500	0,37	0,37	0,38	0,37
	250000	15,79	17,34	15,72	18,69
6	2076834	168,54	169,41	172,69	169,99
	2500	0,36	0,42	0,38	0,47
	250000	15,76	18,88	15,72	16,16
8	2076834	169,55	174,27	170,92	167,49
	2500	0,42	0,35	0,36	0,45
	250000	15,71	15,62	15,7	15,97
12	2076834	168,79	168,44	166,49	170,4

Tabela 3: Estimação do Tiling level

O *script* SQL, para criar o índice QuadTree.

```
CREATE INDEX quadtree_idx ON
FMA (coluna do tipo geometria)
INDEXTYPE IS MDSYS.SPATIAL_INDEX;
PARAMETERS ('sdo_level=yy sdo_numtiles = xx')
```

SQL 5: Código SQL para criar índice QuadTree

3.3.2 Código SQL de pesquisa

Conforme foi mencionado anteriormente no Capitulo 2, as operações a serem realizadas, são operações topológicas, que incluem cálculos de vizinhança local entre pontos e conjunto de pontos. A *Oracle Spatial* já traz estas funções integradas na componente espacial facilitando assim a manipulação dos dados armazenados.

Nesta pesquisa procurou-se identificar todos os pontos que estão á uma distância *d* de um ponto de pesquisa *p.* Este tipo de pesquisa no Oracle Spatial foi realizada através do operador *SDO_GEOM.SDO_BUFFER (geometria, distância, precisão unidade de medida).* Esta função constrói um buffer em volta do ponto *p,* que determina quais os pontos estão dentro do raio de pesquisa (ver Figura 10). Neste teste pretendeu-se calcular todos os pontos que estão a r metros do ponto *p.* Também foi utilizada a função *SDO_ANYINTERACT* que devolve os pontos que interceptam a *query* de pesquisa.

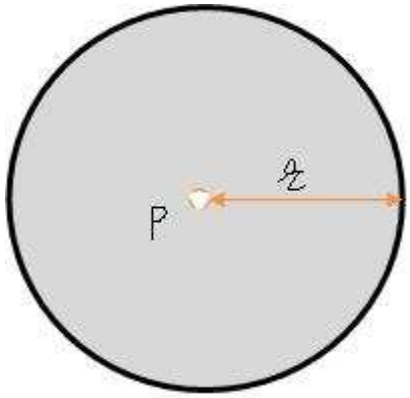

Figura 10: Ilustração do funcionamento do operador SDO_GEOM.SDO_BUFFER

Na caixa de texto em baixo é apresentado o código SQL utilizado para devolver apenas o Identificar (ID) de todos os pontos que estão a uma distância *r* de um ponto p de pesquisa.

Além desta função podia-se usar o *SDO_WITHIN_DISTANCE,* que calcula a distância euclidiana entre em duas geometrias. Sendo que o resultado são os memos que o utilizado.

```
SELECT fma.id as ID
FROM FMA fma
WHERE SDO_ANYINTERACT
(
      fma.pontos,
      SDO_GEOM.SDO_BUFFER(mdsys.SDO_GEOMETRY(2001, 2032629,
```

```
        SDO_POINT_TYPE(x_SQL,y_SQL, NULL), NULL,
        NULL),raio,0.001,'UNIT = METER')
) = 'TRUE';
```

SQL 6: Código SQL de pesquisa dos pontos que estão definidos dentro de um buffer de pesquisa.

3.4 Integração do FMA com a base de dados espacial

Para conectar o Microsoft Visual Studio (VS.Net) com o Oracle foi utilizado a componente ODP.Net (*Oracle Data Provider. Net*). Esta componente disponibilizada pela Oracle permite qualquer aplicação . *Net* efectuar ligação a uma base de dados Oracle. A Figura 11 ilustra um modelo de arquitectura entre o Vs.Net e o Oracle.

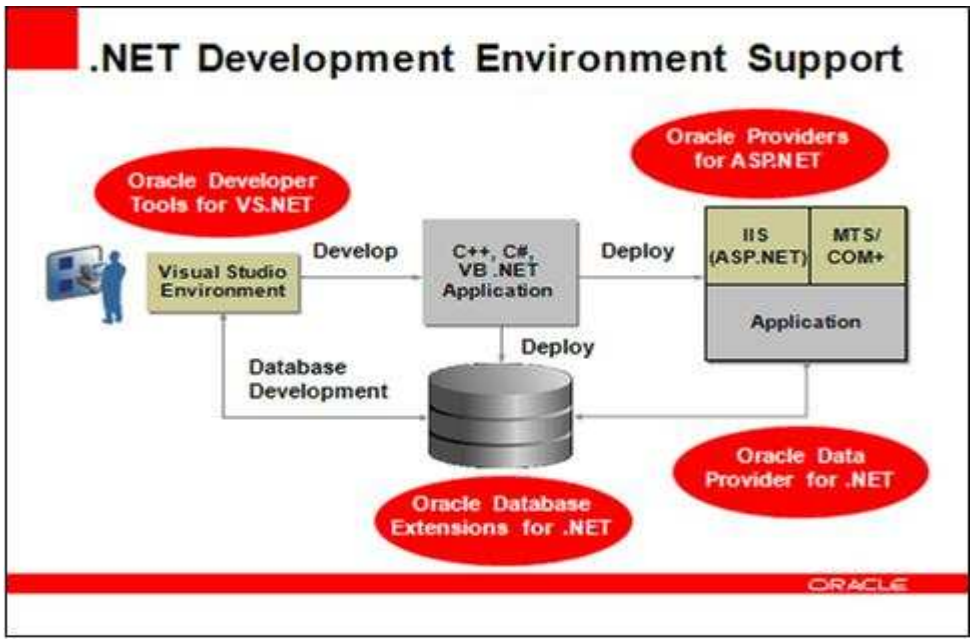

Figura 11: Oracle &. Net, fonte (Oracle, 2007)

Na base de dados foi criado uma *package* que utiliza o procedimento ilustrada na secção 3.5.1para devolver todos os pontos que estão a uma distância *raio* de um ponto de pesquisa. As variáveis de pesquisa são enviadas á Base de dados para o cursor, e é retornado o *id* dos pontos que satisfazem o critério de pesquisa. Foi adoptado o retorno apenas do identificador do ponto e não as respectivas coordenadas dado que no início da execução do programa foi carregado todos as coordenas para uma estrutura em

memória. Desta forma reduz-se o volume de informação a transferir entre a base de dados Oracle e aplicação *VS.Net*.

```
CREATE OR REPLACE PACKAGE cursor_fma AS
        TYPE t_cursor IS REF CURSOR;
 Procedure open_join_cursor (x_SQL IN NUMBER, y_SQL IN NUMBER,
raio IN NUMBER, tab IN VARCHAR, io_cursor IN OUT t_cursor);
   END cursor_fma;
  /

CREATE OR REPLACE PACKAGE BODY cursor_fma AS
   Procedure open_join_curosor(x_SQL IN NUMBER, y_SQL IN NUMBER,
raio IN NUMBER, io_cursor IN OUT t_cursor)
   IS
        v_cursor t_cursor;
   BEGIN
             OPEN v_cursor FOR
               SELECT fma.ID
                FROM FMA fma
                WHERE SDO_WITHIN_DISTANCE (fma.PONTOS,
mdsys.SDO_GEOMETRY(2001, 2032629,
                SDO_POINT_TYPE(x_SQL,y_SQL, NULL)
              , NULL, NULL),'distance = raio unit=meter') =
'TRUE';
        io_cursor := v_cursor;
   END open_join_cursor;
   END cursor_fma;
  /
```

SQL 7: Package Pl/SQL para devolver o identificador dos pontos que satisfazem um critério de pesquisa

Criando esta *package* na base de dados, no lado do *VS.Net* apenas é preciso enviar os parâmetros para fazer o *bind* na base de dados. Desta maneira consegue-se ganhar 50% de tempo ou mais, caso esses parâmetros estavam a ser enviados através de uma *string* normal para a BD utilizando o *OracleCommand*.

A seguir é apresentado o diagrama de arquitectura do modelo de conexão de dados fornecido pela ODP.NET. Este modelo de conexão de dados, visa sobretudo trabalhar com os dados em memória, já que muitos acessos aos discos podem não ser a opção acertada para aumentar o desempenho.

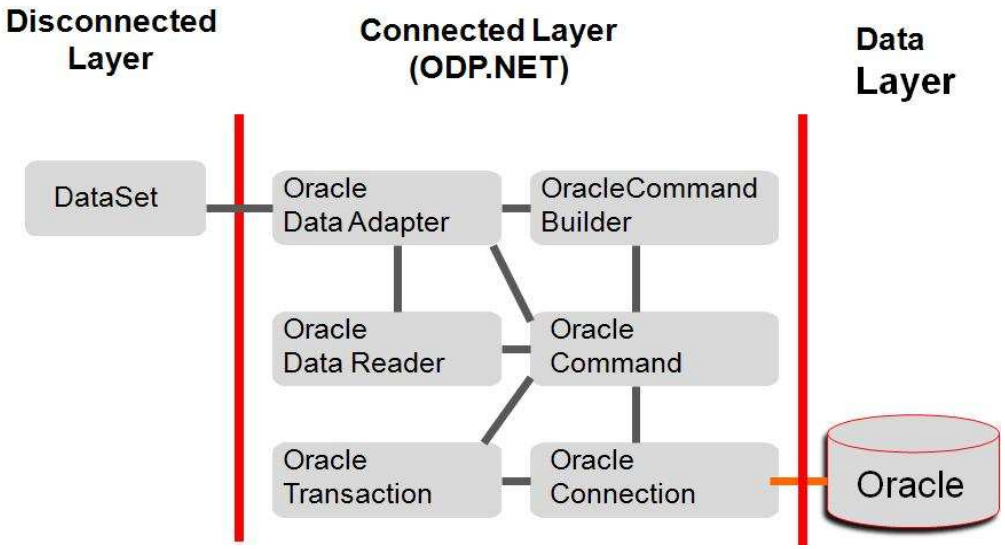

Figura 12: Modelo de conexão de dados, fonte (Oracle, 2007)

Este tipo de arquitectura visa sobretudo evitar muitos acessos aos discos e aumentar a performance no acesso aos dados. Ao conectar-se à base dados, o resultado de execução da pesquisa é armazenada temporariamente na memória e é interrompida a conexão com a base de dados. As operações de manipulação dos dados são realizadas sobre a informação alocada em memória. Por fim, quando não se pretende efectuar mais alterações, a conexão é restabelecida e as alterações gravadas na base de dados.

3.5 Análise comparativa dos métodos de indexação

Para avaliar as estruturas de dados, o tempo de processamento e o acesso eficiente da informação espacial foi o principal critério de medida. Outros parâmetros como o espaço de disco necessário para armazenar os índices não foram contabilizados para a análise das estruturas.

- O tempo de processamento
  - Tempo de criação do índice
  - Tempo de processamento discriminada por iteração em duas RAM
  - Tempo de execução das pesquisas executadas directamente na base de dados

Cada RAM corresponde a uma pesquisa executada sobre o *script* (*SQL6*) apresentada na secção 3.5.6.

Além dos testes apresentados em cima, realizados com vista a verificar o comportamento de cada estrutura mediante o aumento da dimensão dos dados e aumento do *kernel* de pesquisa, foi realizado um teste experimental, que consistiu na comparação do tempo do processamento, na filtragem do algoritmo original sem indexação espacial e o algoritmo otpimizado já aplicado o conceito de indexação espacial.

Todos os testes foram realizados no sistema operativo Microsoft XP Professional x64, instalado num computador com um processador Intel Xeron (R), E5335 de oito núcleos a 2.00 GHz, com 4.00 GB de memória RAM e 1 TB de disco rígido.

Capítulo 4

4. Resultados e Discussão

Conforme citado no Capítulo 3, as estruturas de indexação espaciais são utilizadas para armazenar grandes volumes de dados, e organizá-los num espaço multidimensional de forma, a que apenas um subconjunto de objectos é utilizado para dar resposta ao pedido, aumentando assim o desempenho no processamento dos algoritmos que baseiam, em operações de vizinhança local.

Para a obtenção de um resultado credível, foi feita uma comparação num ambiente homogéneo, seguindo a filosofia que este ambiente deve utilizar o mesmo conjunto de dados, proporcionar o mesmo ambiente de desenvolvimento e medir as mesmas variáveis.

Neste Capítulo são apresentados os resultados globais obtidos da optimização da FMA realizado em memória e usando acesso aos discos.

4.1 Método de optimização do FMA

Na Tabela 4 estão representados os resultados obtidos da optimização do FMA utilizando apenas estruturas de dados armazenados em memória.

	Tempo processamento(s)			
	2500	25000	50000	200000
Algoritmo Original	2,52	230,24	968,21	22951,83
Algt. Orig. Optimizado	12,07	174,25	384,93	2533,34

Tabela 4: Resultado das optimizações realizadas em memórias

Como se pode constatar com a Figura 13, a solução optimizada apresenta um factor de crescimento consideravelmente menor, que se deve justamente à diminuição

considerável do número de operações de vizinhança local, proporcionada pelo método de pré-processamento proposto.

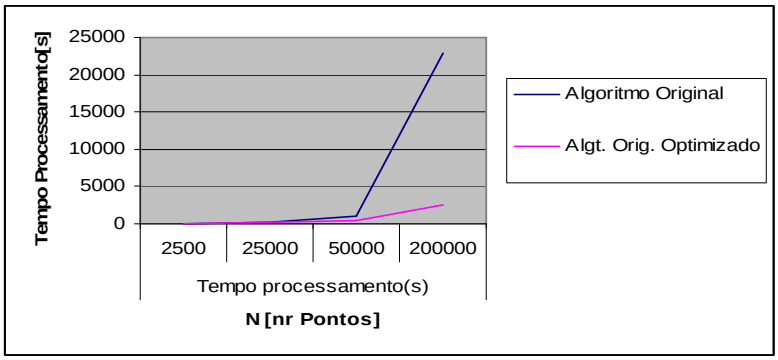

Figura 13: Tempo execução da solução original e optimizada.

No entanto, a quantidade de dados que se conseguem filtrar está limitado à quantidade de memória RAM disponível na máquina onde o FMA é executado (informação mais detalhada sobre o resultado dos testes consultar o Anexo IV).

4.2 Avaliação das estruturas na base de dados

4.2.1 Criação de índice

Na tabela 5 estão ilustrados os tempos de criação de índice para a R-Tree e QuadTree (*tiling level = 12* e número de *tiles* = 100000). Como se pode observar não existem diferenças significativas com o aumento da dimensão dos dados.

Dimensão	R-Tree	QuadTree
2500	0.38	0.45
250000	15.9	15.97
2076834	177.5	170.4

Tabela 5: R-Tree: Tempo de criação do índice

4.2.2 Avaliação do *script* de pesquisa

Nesta comparação pretendeu-se estimar o tempo de processamento das pesquisas realizadas na base de dados em duas RAM, fazendo variar o raio de e o volume dos dados avaliado em números de pontos, para o *script* de pesquisa *SQL6*.

		Tempo Processamento(s)			
		R-Tree		QuadTree	
raio(m)	nrº pontos	RAM1	RAM2	RAM1	RAM2
	2500	0,13	0.07	0.10	0.05
50	250000	0,13	0.05	0.10	0.03
	2076803	0,33	0.26	0.31	0.12
	2500	0.12	0.03	0.63	0.08
500	250000	0.15	0.05	0.11	0.05
	2076803	0.23	0.06	0.13	0.05
	2500	0.13	0.03	0.11	0.05
1000	250000	0.12	0.05	0.10	0.05
	2076803	0.14	0.03	0.15	0.06

Tabela 6: Tempo processamento variando o raio de pesquisa em duas RAM

Como se pode verificar pela Tabela 6 o tempo de processamento nas duas RAM é bastante diferente. Esta diferença deve-se ao facto da estratégia de filtragem executa por cada um dos métodos (para mais detalhes sobre o processamento de pesquisas espaciais no Oracle ver o Anexo I.V). O objectivo deste teste foi verificar se existia um efeito cache nas pesquisas, no entanto, constatou-se que esta variação está mais ligada às técnicas de filtragem executa pela *Oracle Spatial*. No caso deste estudo é irrelevante dado que trabalhamos apenas com uma filtragem

4.3 Processamento do Algoritmo por Iterações

Nesta secção serão avaliados os tempos de processamento para verificar o comportamento dos métodos de indexação já integrados no FMA. Além do tempo total, foi analisado o tempo de processamento por cada iteração nas operações de Abertura (ver Tabela 7).

		R-Tree		QuadTree	
Iteração	nrº pontos	Erosão	Dilatação	Erosão	Dilatação
	2500	66,92	0.08	64,03	0.0
1	25000	645,83	0,09375	654,31	0,09
	2500	70,44	0.03125	67,25	0.09
2	25000	682,23	0,375	656,06	0,38
	2500	82,69	0.09375	70,09	0,09
3	25000	808,672	1,2	778,56	1,21
	2500	118,88	0.32	112,14	0,32
4	25000	12279,23	4,31	1227,18	4,31
	2500	141,23	8,14	135,84	0,5
5	25000	1685,81	6,26	1615,75	6,23

Tabela 7: Tempo de processamento discriminada por iteração

Na Figura 14 e 15 são ilustrados a evolução dos tempos de processamento por iterações, da R-Tree e da QuadTree, para distintas dimensões.

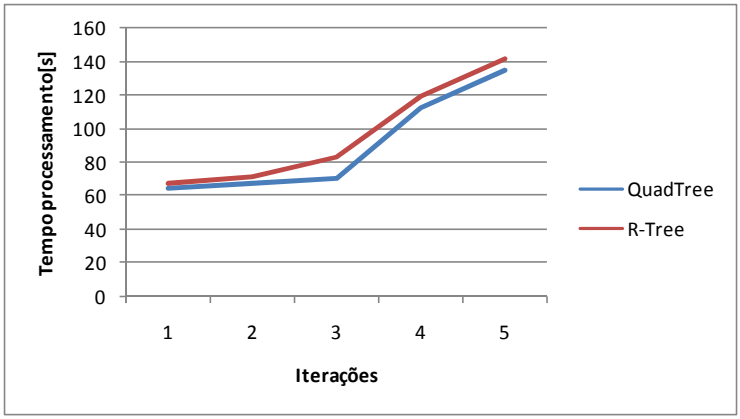

Figura 14: Tempo de processamento por iteração para 2500 pontos

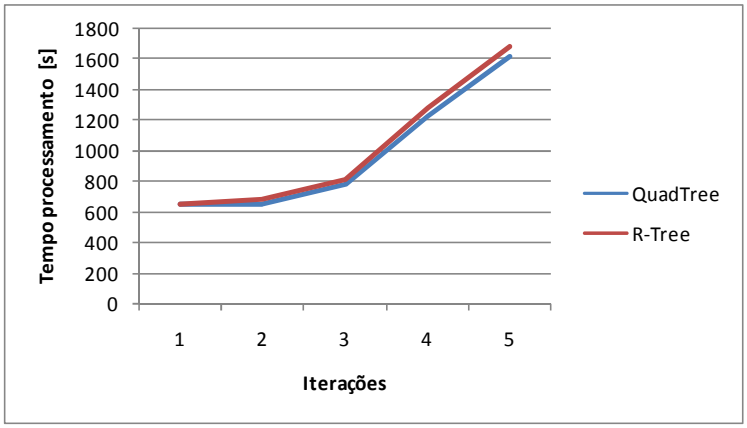

Figura 15: Tempo de processamento por iteração para 25000 pontos

Na avaliação da R-Tree e QuadTree como se pode verificar nos testes apresentados para pequenas quantidades de dados a QuadTree tende a apresentar melhor desempenho. No entanto, com o aumento da dimensão dos dados esta tendência tende-se a inverter. A mudança deve-se sobretudo à estratégia de busca realizada pela R-Tree. Em ambos os métodos numa primeira fase é aplicado um filtro primário para seleccionar as geometrias candidatas ao *script* de pesquisa. Essas geometrias são enviadas para um filtro intermédio onde ao ser retornado *true* a geometria deverá ser incluída no resultado final; *false* caso contrário; e *unknown* caso não for possível determinar logo o resultado da operação, sendo então necessário aplicar um filtro secundário para determinar as relações exactas das geometrias. Para determinar essas relações, o filtro intermédio utiliza o interior da *query* de pesquisa e as geometrias candidatas (recebidas do filtro

primário). No caso da QuadTree o interior e os limites de *tiles* das geometrias candidatas são comparados com o interior e os limites da geometria da *query* de pesquisa. No caso da R-Tree, apenas é analisado o interior da geometria de pesquisa. Para a QuadTree como já se sabe ao definir o *tiling level*, cada geometria é aproximada por um conjunto mínimo de tiles que o engloba. Quanto maior o *tiling level*, mais precisas são as aproximações das geometrias, consequentemente mais refinadas são as buscas. Mais informações sobre técnicas de filtragens utilizadas por cada um dos métodos no *Oracle Spatial* podem ser encontradas no ANEXO III.

4.4 Avaliação das optimizações realizadas com acesso aos discos

A seguir são apresentados os resultados do tempo de processamento da solução optimizada com acesso aos discos. O tempo de processamento inicial, necessário para calcular as distâncias pelo maior *kernel* a partir do Oracle é apresentado antes do início das iterações.

	R-Tree				QuadTree			
	Dimensão(nr ºpontos) / Tempo(s)							
Iteração	2500	25000	50000	200000	2500	25000	50000	200000
Pré. Process.	126,16	1241,04	2555,2	12423,48	117,18	1278,69	2557,85	12249,73
1	0,31	3,17	6,3	48,69	0,3	3,17	6,81	49,64
2	0,34	3,63	7,09	51,89	0,33	3,63	7,88	53,13
3	0,5	5,36	10,63	56,24	0,5	53,28	11,39	56,69
4	1,08	11,92	24,13	126,21	1,08	118,29	24,88	127,27
Total	128,39	1265,12	2603,35	12706,5	119,38	1457,05	2608,8	12536,46
Média por Pontos	0,05	0,05	0,05	0,06	0,05	0,05	0,05	0,06

Tabela 8: Tempo de processamento solução optimizada com acesso aos discos

Na tabela 9 em conjunto com a Figura 14 são representados as evoluções do tempo de processamento para as diferentes soluções.

	Tempo total do processamento(s)			
	2500	25000	50000	200000
Algt. Orig. Optmizado	12,07	174,25	384,93	2533,34
Algt. Final QT	119,38	1457,05	2608,8	12536,46
Algt. Final RT	128,39	1265,12	2603,35	12706,5
Algt. Original	2,52	230,24)	968,21	22951,83

Tabela 9: Tempo de processamento total discriminada por algoritmos

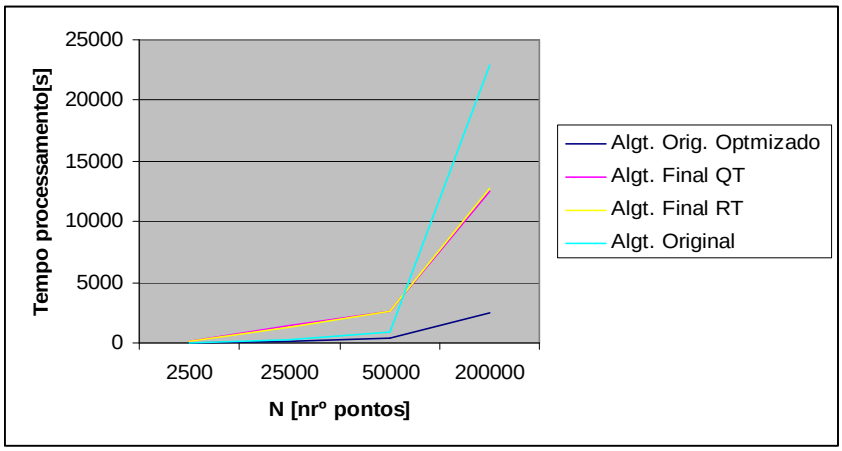

Figura 16: Tempo processamento para as diferentes soluções

Na Figura 16 pode-se verificar que foram desenvolvidas mais três versões além da solução original; Uma solução optimizada em memória da solução original representada a roxo, e duas soluções a amarelo e rosa, utilizando estruturas de indexação espacial R-Tree e QuadTree, respectivamente.

Na solução original e optimizada, inicialmente são carregados os dados para estruturas de dados armazenados em memória, sendo a posterior manipulação realizada sobre esta estrutura. As soluções com índices espaciais optimizam o acesso à informação espacial mas utilizando índices armazenados em disco.

Verificou-se que para pequenas dimensões de dados a solução original apresenta melhor resultado do que as soluções optimizadas em disco. À medida que o volume de dados aumenta o tempo de processamento da solução original não optimizada crescia numa escala superior em relação às soluções optimizadas em disco. Na tabela 9, pode-se verificar que de 2500 para 200000 pontos, no algoritmo original o tempo de processamento aumenta 11000 vezes, no optimizado em memória 200 vezes e nos optimizados com índice em disco 100 vezes (notando uma certa “linearidade” com o aumento do número de pontos).

Esta variação do tempo de processamento com o aumento dos dados deve-se à não utilização de mecanismos eficientes de indexação, obrigando a realizar qualquer pesquisa sobre a base de dados inteira. Exemplificando para realizar uma simples pesquisa de vizinhança local como encontrar todos os pontos que estão a uma distâncias *d*, de um ponto de pesquisa *p* é necessário percorrer a base de dados inteira para verificar apenas os pontos que estão dentro da distância de pesquisa pretendida.

Nas soluções optimizadas, os métodos de acesso multidimensionais, organizam o espaço multidimensional e os objectos contidos sobre ela, de tal forma que somente

algumas partes do espaço e um subconjunto dos objectos espaciais armazenados sejam utilizados para dar resposta ao pedido realizado.

Capítulo 5

5. Conclusões e Perspectivas de Desenvolvimento

5.1 Conclusões

No decurso desta dissertação foi realizado um estudo comparativo de métodos de indexação espacial que possibilitassem a representação dos dados LIDAR num espaço multidimensional, assim como, a realização de pesquisas de vizinhança local de forma eficiente.

Dos métodos analisados fez-se uma selecção com base na literatura científica, acabando por ser seleccionado a QuadTree e a R-Tree, representando a família dos métodos baseados na decomposição recursiva do espaço e baseada na distribuição espacial dos objectos respectivamente.

O algoritmo de filtragem baseado em morfologia matemática foi implementado utilizando a linguagem de programação C#, com base numa solução existente desenvolvida em IDL., mostrando esse adequado para o problema em causa.

Dos Sistemas de Gestão de Base de Dados (SGBDs) que oferecem extensão espacial a Oracle é a que disponibiliza funcionalidades mais completas para a manipulação de dados espaciais e já inclui as duas estruturas de indexação seleccionadas. Sendo assim todos os testes de avaliação de desempenho das duas estruturas, passavam a ser realizadas no mesmo ambiente de desenvolvimento e com os mesmos parâmetros contribuindo assim para uma tomada de decisão mais acertada.

Mediante os testes realizados para avaliação dos índices espaciais, apesar da QuadTree apresentar valores ligeiramente melhores que a R-Tree esses não são relevantes.

A solução original usa estrutura de dados armazenados em memória. Enquanto as soluções com índices espaciais optimizam o acesso aos dados, mas utilizando índices

armazenados em disco. Constatou-se que apesar da solução original não optimizada estar a executar em memória, a solução implementada com acessos ao disco começa a apresentar melhor resultado no intervalo de] 50; 200000] pontos. Verificou-se ainda que de 2500 para 200000 pontos a solução original não optimizada aumenta cerca de 11000 vezes, na optimizada em memória 200 vezes e nas optimizações em disco 100 vezes.

Com maior ou menor grau de dificuldade os objectivos foram cumpridos deixando sempre soluções plausíveis de optimização para futuros trabalhos.

5.2 Perspectivas de Desenvolvimento

A realização desta dissertação foi um desafio pessoal, na medida que o autor não tinha nenhum conhecimento nas áreas das Ciências da Terra e do Espaço e da Engenharia Geográfica. Nesta primeira investigação não se conseguiram extrapolar conclusões objectivas, mas sim algumas tendências que serão úteis na condução de uma tese de doutoramento ao qual pretendo realizar.

Dado a elevada complexidade do problema em si, esta dissertação serviu para assimilar novos conceitos até então desconhecidos, que irão permitir em futuras investigações a delineação de caminhos mais sustentados para a evolução da mesma. Sendo assim nesse sentido entende-se que as seguintes orientações que iremos levar a cabo, deverão passar por investigações mais profundas de análise de processamento contemplando outras estruturas de indexação espaciais, assim como outras variáveis que contribuem para avaliar a performance do algoritmo.

Finalmente tem-se como objectivo desenvolver de raiz um método eficaz que permitam optimizar este tipo de filtragens de modo a optimizar o tempo de processamento de forma significativa. Posteriormente pretendemos explorar as potencialidades do paralelismo no processamento a partir de um modelo já optimizado.

Outra optimização que pode ser realizada tem a ver com a estratégia de cache espacial, de maneira a permitir manter me memória os objectos espaciais, ajudando a reduzir as operações de buscas desses objectos do disco para a memória.

Referências Bibliográficas

Ackermann, 1999. Airborne laser scanning—present status and future expectations. ISPRS Journal of Photogrammetry and Remote Sensing Volume 54, Issues 2-3, July 1999, Pages 64-67.

Ahokas, E., Kaartinen, H., y Hyyppa, J. 2003. *"A quality assessment of airborne laser scanner data"*. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*; Vol. XXXIV (3/W13). Dresden, Alemania.

Arefi, H., Hahn, M., y Lindenberger, J. 2003. *LiDAR data classification with remote sensing tools.* Joint ISPRS Comission IV Workshop "Challenges in Geospatial Analysis, Integration and Visualization II'', Stuttgart, Alemanha.

Axelsson, P. 2000, *DEM generation from laser scanner data using adaptive TIN models*. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences; 34 (B4/1): 110 - 117.

Baltsavias, E. P. 1999. *Airborne laser scanning - relations and formulas*. ISPRS Journal of Photogrammetry and Remote Sensing; 54: 199 - 214.

Beomseok, Nam; Alan, Sussman: *A comparative study of spatial indexing techniques for multidimensional scientific datasets.* Proceedings 16th International Conference Volume, Issue, 212-213 June 2004, pages 171-80

BORGES, 1977. Topografia. São Paulo: Edgard Blüncher.Chrisman, N., 1984. *The role of quality information in the long-term functioning of a geographic information system.* Cartographica 21, pp. 79–87.

CIFERRI, R. SALGADO, A. C: Factores *Determinantes de Desempenho de Métodos de Acesso Multidimensionais.* In: II Workshop Brasileiro de Geoinformática - GeoInfo2000, São Paulo 2000, p. 112-119.

Crombaghs, M.J.E., Oude Elberink, S.J., Brügelmann, R., de Min, E.J., 2002. *"Assessing height precision of laser altimetry DEM's"*. In: International Archives of Photogrammetry and Remote Sensing, vol. 34, part 3A "Photogrammetric Computer Vision", Graz 2002, pp. 85-90.

Donald Knuth: *The Art of Computer Programming*, Volume 3: *Sorting and Searching*, Third Edition. Addison-Wesley, 1997

D. J. Abel: Bit-interleaved keys as the basis for spatial access in front-end spatial database management system. In: B. Diaz, S. Bell, (eds): *Spatial Data Processing Using Tesseral Methods,* Natural Environment Research Council, 163-177, 1986

David. Déharbe: Elementos *da complexidade computacional,* DIMAp/UFRN, Agosto de 2005.

Finkel, R.A; Bentley, J.L QuadTree: *A data structure for retrieval on composite keys.* Acta Informática p. 1-9, 1974

Flood, M., 2001. "*Lidar activities and Research priorities the commercial sector*". In: IAPRS, Annapolis, America, Vol. XXXIV-3/W4, pp. 3-7.

Fowler, R., 2001. "Topographic lidar". In: D.F. Maune (ed.), *Digital Elevation Model Technologies and Applications*. American Society for Photogrammetry and Remote Sensing, Bethesda, Maryland, pp. 207–236.

Gomes Pereira, L.M., Wicherson, R.J. 1999. "*Suitability of laser data for deriving geographical information: a case study in the context of management of fluvial zones*". ISPRS Journal of Photogrammetry and Remote Sensing, 54 (2-3), pp. 105-114.

Gonçalves-Seco, L. *Automatización en la generación de modelos digitales de elevación, cubicación y combustibles, a partir de LiDAR.* Ph.D Disertación, Universidad de Santiago de Compostela; 275 pp.

Gonçalves-Seco L., González Ferreiro, E., Diéguez-Aranda, U., Crecente Maseda, R., Miranda Barrós, D. 2007a. *"Automated estimation of stand volume in eucalyptus globulus plantations using airborne laser scanner data"*. Proceedings of ForestSat'07, 7 pp. Montpellier (France).

Gonçalves-Seco, L., Miranda, D., Crecente, R., 2006. "*Digital Terrain Model generation using airborne LiDAR in a Forested area of Galicia (Spain)*". Proceedings of the Seventh International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences. Lisbon (Portugal), pp 169-180.

Gonçalves-Seco L., Fraga-Bugallo, B., Crecente Maseda, R., Miranda Barrós, D. 2007b. *"Automatic extraction of forest and terrain fuel variables with airborne lidar to determine structural fire risk*". Proceedings of ForestSat'07, 9 pp. Montpellier (France)

G, Kedem: The Quad-CIF Ttree*: A data structure for hierarchical on-line algorithms,* In ACM IEEE Nineteenth Design Automation Conference Proceedings, p. 352–357, Los Alamitos, Ca., USA, June 1982.

G, Peano, *Sur une courbe, qui remplit toute une aire plane.* Math. Ann 36, pages 157-160, 1890

Gonçalves-Seco, L., 2007. *Utilização de Dados LIDAR na Classificação de Objectos Situados em Zonas Rurais*. V Conferência Nacional de Cartografia e Geodesia. Lisboa (Portugal). 19-20 Abril.

Isaaks, E. H.; Srisvatra, R. M. *An introduction to applied geoestatistics.* New *York:* Oxford University, 1989. Page 561.

J. L. Bentley. *Multidimensional binary search trees used for associative searching.* In Communications' of the ACM 18 (9), 1975

J. Gray and G. Graefe: *The five-minute rule ten years later, and other computer storage rule of thumb*. In Proceedings of the 1997 ACM SIGMOD International conference on Management of Data, volume 26, p. 63-687, 1997

J. Nievergelt, H. Hinterberger, K. C. Sevcik, The grid file: *An adaptable, sysmetric multikey file structures.* ACM Transactions on Database, pages 1, 38-71 1990

Maas, H.-G., 2002. *"Methods for measuring height and planimetry discrepancies in airborne laserscanner data"*. Photogrammetric Engineering and Remote Sensing 68 (9), pp 933–940.

Lefsky MA, Harding D, Cohen WB, Parker G, Shugart HH 1999. "Surface LiDAR remote sensing of basal area and biomass in deciduous forests of Eastern Maryland, USA". Remote Sens Environ 67, pp 83–98

Mark Allen Weiss: *Data Structures & Algorithm Analysis in java,* Addison- Wesley Publishing Company, Reading, MA, p.35-48

Næsset, E. 2004. "Practical large-scale forest stand inventory using a smallfootprint airborne scanning laser". Scandinavian Journal of Forest Research, 19, pp 164−179.

N. Katayama and S. Satoh. *The SR-Tree: An index structure for high dimensional nearest neighbor queries*. In Proceedings of the 1997 ACM SIGMOD International conference on Management of Data, volume 26, Pages 369-380, 1997

Neto. 1998, *Sistemas de Informação Geográfica*. 2 Ed. FCA – Editora de Informática LDA; 1998

Neteler, Markus & Mitasova, Helena. 2003. Open Source GIS: A GRASS GIS Approach. 2 ed. Kluwer Academic Publishers;

N. Beckman, H. P; Kriegel, R. Schneider and B. Seeger. The R*-Tree: *An efficient and robust access method for points and rectangles.* Proceedings of the ACM SIGMOD International Conference on the Management of Data, 1990, p.322-31

Nakamura, S. Abe, Y. Ohsawa, M.Sakauchi, *A Balanced Hierarchical Data Structure for Multidimensional Data with Highly Efficient Dynamics Characteristics,* IEE Transactions on Knowledge and Data Engineering, v.5 n.4, o.682-649, August 1993

Kraus, K., Pfeifer, N., 1998. *"Determination of terrain models in wooded areas with airborne laser scanner data"*. ISPRS J. Photogramm. Remote Sensing 53 ,4., pp 193–203.

Oracle SPATIAL 11g: *Advanced Spatial Data Management for the Enterprise.* Março 2007,

[ORA08] URL: *http://www.oracle.com/technology/index.html.* consultado em Março 2008

Pereira, L. 2005. *"Varrimento aéreo por laser: princípios e estado da arte"*. Actas da IV Conferência Nacional de Cartografia e Geodesia. Lidl, Alemania; pp 313-323.

Petzold, B., Reiss, P., & Stossel, W. 1999. *"Laser scanning—surveying and mapping agencies are using a new technique for the derivation of digital terrain models"*. International Society of Photogrammetry and Remote Sensing Journal of Photogrammetry and Remote Sensing, 54, pp 95– 104.

Ravi, Kothuri; Albert, Godfrind; Euro, Beinat: *Pro Oracle Spatial Database 11g* *Apress*, p. 252-255

Riaño, D., Meier, E., Allgöwer. B., Chuvieco, E., Ustin, S. L., 2003. *"Modeling airborne laser scanning data for the spatial generation of critical forest parameters in fire behavior modeling*. Remote Sensing of Environment 86, pp 177–186

Samet, H: *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, 1990.

Sapeta, K: *"Have you seen the light?" LIDAR technology is creating believers.* Geoworld, http://www.geoplace.com/ME2/Default.asp, ultimo acesso Abril de 2008

Sellis, Timos; Roussopouls, Faloutos Christos: *The R+-Tree: A Dynamic Index for Multi-Dimensional Objects 13th* VLDB Conference, p 507-517 Brighton 1987

Sithole, G., 2005. "*Segmentation and Classification of Airborne Laser Scanner Data*". Ph.D. Dissertation, Technische Universiteit Delft; 204 pp.

S. Berchtold; D. A Keim; and H. Kriegel: The X-Tree*: An index structure for high .dimensional data*. Proceedings of the 22nd International Conference on Very Large Data Bases, Mumbai (Bombay) 1996.

Song, J. H., Han, S. H., Yu, K. Y., y Kim, Y. I. 2002. Assessing the possibility of land-cover classification using LIDAR intensity data. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences; Commission III Symposium. Graz, Austria.

Stan Aronoff, "Remote Sensing for GIS managers", ESRI press, Redlan

Thomas, H. Meyer: *Fast Algorithms using minimal Data Structures for common topological relationships in large, irregularly spaced topographic data sets.* Computer & Geosciences, 2006

Hanan, Samet. *Intelligence Foundations of Multidimensional and Metric Data Structures*, Addison- Wesley Publishing Company, Reading, MA, p. 507

Haralick, R.M. and Shapiro, L.G: A Survey: *Image Segmentation Techniques, Journal* of Computer Vision, Graphics and Image Processing, p. 103-132, 1985

Hyyppä, J., Hyyppä, H., Litkey, P., Yu, X., Haggrén, H., Rönnholm, P., Pyysalo, U., Pitkänen, J., Maltamo, M., 2004. *"Algorithms and methods of airborne laser-scanning for forest measurements"*. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 36 (Part 8/W2), pp 82–88.

White, D; A. Jain, R.: *"Similarity Indexing with the SS-tree"*, Proc. 12th Int. Conf. On Data Engineering, New Orleans, LA., Jan. 1996--pp. 516

Wehr, A; Lohr, U. *Airborne laser scanning – an introduction and overview*. Isprs Journal of Photogram metric and Remote Sensing, pages 68-82, 1999.

Worboys, M.F. and Duckham, M. (2004) GIS: A Computing Perspective, Second Edition, CRC Press, ISBN: 0415283760.

Xiao, Weiqi; Feng Yucai, Geosciences and remoting sensing: *A Scientific Vision for Sustainable Development*, 1997 IEEE International Volume 1, pages 216-218

Zang, W and D.R. Montgomery, 1994, *Digital Elevation Model Grid Size, Landscape Representation and Hydrologic Simulations*, Water Resources Research, 30 (4):1019-1028.

ANEXO 1: A Tecnologia LLIDAR

Recentemente a tecnologia LIDAR tem evoluído de uma forma notável, transformando-se num instrumento eficiente e de baixo custo, tendo inúmeras aplicações actuais, entre as quais se destaca a Gestão Florestal, Planeamento Urbanístico, Projectos Arquitectónicos e Paisagísticos, aplicações CAD/ CAM (projecto e fabricação auxiliada por computadores) e aplicações de biotecnologia.

Os dados LIDAR são recolhidos através de sensores colocados estrategicamente em aviões, helicópteros ou em aeroplanos a vários metros de altitude. A nuvem irregular dos pontos 3D (pontos georreferenciados), permite criar o modelo da superfície do terreno em função dos retornos registados para cada pulso (normalmente primeiro e ultimo), e é possível distinguir os terrenos (MDT) e as elevações (Modelo Digital de Superfície – MDS);

Funcionamento do LIDAR

O LIDAR aerotransportado aplicado para fins cartográficos é um sistema activo baseado num sensor laser. Consiste na emissão de um pulso de laser e na medição do tempo de retorno que demora a chegar à superfície terrestre e ser devolvida de novo ao avião.

LIDAR funciona como um radar ordinário, que apenas emite luz em vez de ondas de rádio [WL99]. O scanner esta constituído por um transmissor laser que cria pulsos ópticos. Estes pulsos ao entrar em contacto com um objecto, são reflectidos e recolhidos através de um sensor óptico-electrónico.

A velocidade da luz é conhecida, e um contador de alta velocidade mede o tempo de voo desde o inicio do envio do pulso até ao momento de retorno. Por fim, este é convertido numa distância D (desde o scanner até o objecto). Figura 1.

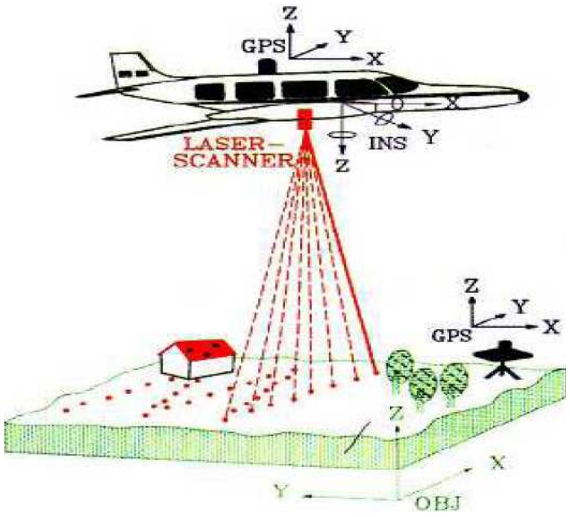

Figura 17: Sistemas de posicionamento num voo LIDAR

Para referenciar correctamente o ponto que foi medido no terreno utiliza-se combinações de duas técnicas diferentes.

- INS (Sistema de Navegação Inercial): Permite medir a orientação exacta do sensor. Este sistema mede os ângulos com uma precisão de 0.001 graus, e permite compensar os movimentos bruscos que o sensor sofre a bordo de um avião, podendo calcular em cada movimento as coordenadas exactas do ponto que este a ser medido.
- GPS diferencial (DGPS) para poder medir a posição exacta do sensor. Um GPS cinemático aerotransportado segue o rastro de pelo menos quatro satélites de navegação e regista a posição espacial do avião.

Quando um raio laser chega ao terreno comporta de uma forma diferente dependendo das características do objecto em que embateu.

1. Numa superfície sólida (edifícios, solo, etc.), o raio laser é reflectido ao avião sem nenhum problema.
2. Em áreas de vegetação, o raio laser choca em primeiro lugar com a copa da árvore. Neste momento parte do raio é reflectida ao avião, dado que se trata de uma superfície sólida, no entanto outra parte do raio que atravessa a vegetação ao embater no solo é devolvido ao avião. Neste caso o sistema guarda o primeiro e ultimo pulso. Figura 2 – b.

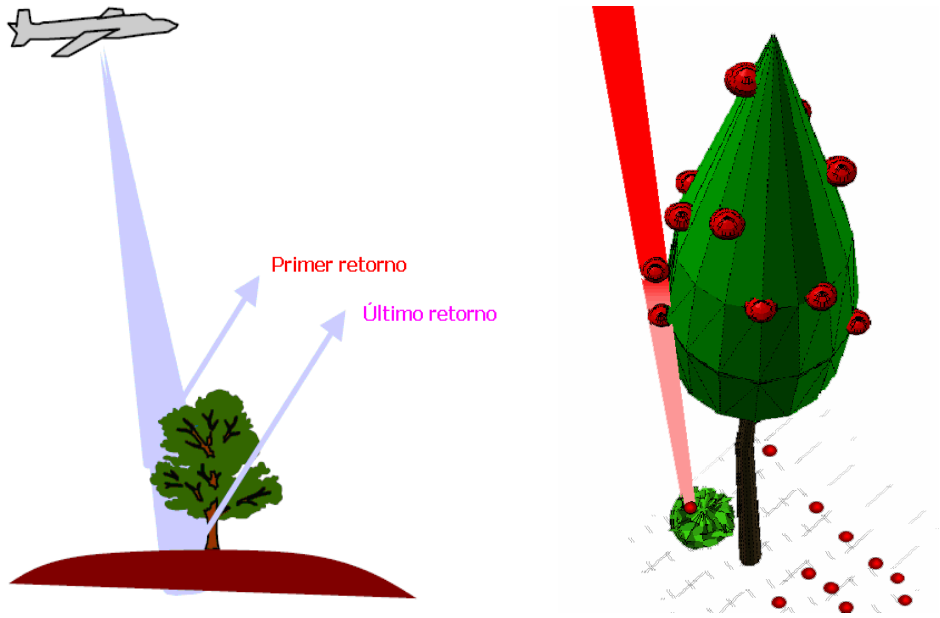

Figura 18: Comportamento do laser ao embater numa árvore

Os dados LIDAR produzidos são combinados com os dados GPS posicionados para georeferenciar o conjunto de dados. Uma vez que os dados do voo são registados e mediante um software apropriado, pode-se passar para a fase de processamento e correcção dos dados, de forma a gerar contornos, modelos de superfície e modelos de elevação.

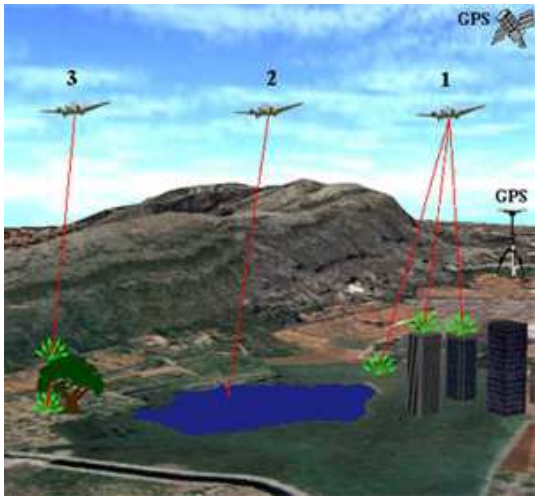

Figura 19: Diferentes características dos objectos e comportamento do laser.

Desta forma se obtêm as alturas do terreno com uma precisão em altura na ordem de 15 cm. Caso o sensor trabalhar com uma frequência 33 kHz, o processo de medida descrito anteriormente se repete umas 33.000 vezes por segundo, pelo que permite obter modelos de alta qualidade, com uma resolução espacial de 1 metro por pixel.

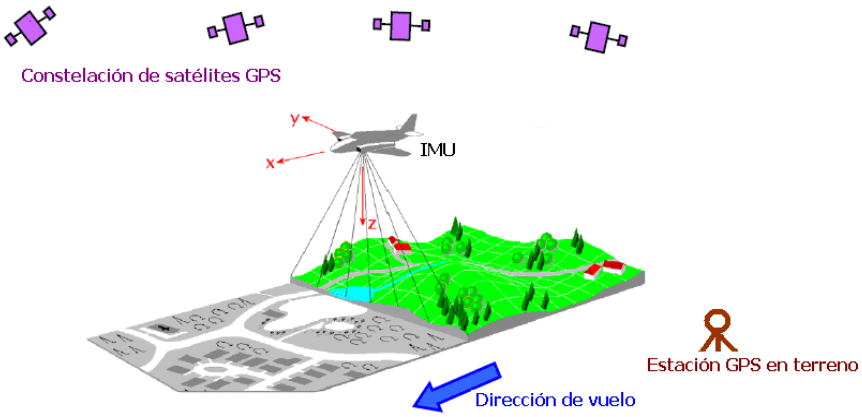

Figura 20: Funcionamento do ALS, fonte: http://www.toposys.de

No processamento de informação 3D, a posição dos pontos se realizam mediante uma unidade de gravação que combina as distancias medidas, o ângulo do espelho, a posição do GPS, a informação que oferece o INS para cada pulso, e realiza uma serie de transformações replicando a rotação e translação do alcance do laser desde as coordenadas local do avião até as coordenadas WGS84 [SK00].

ANEXO 2: Implementação da QuadTree e R-Tree no Oracle Spatial

A R-Tree no Oracle Spatial esta implementada em tabelas da base de dados. As buscas envolvem a utilização de SQL recursivo para se chegar da raiz aos nós da árvore [KT, 2002]. Essa abordagem resulta em buscas mais eficientes devido a uma melhor preservação de aproximações espaciais, mas pode tornar-se lenta para operações de *updates.*

A QuadTree realiza aproximação das geometrias através de aproximações com quadrantes, que são resultantes da subdivisão recursiva do espaço. A QuadTree utiliza índices e *B-Tree* para realizar buscas espaciais e outras operações DML (*Data Manipulating Language*). Essa abordagem acarreta em uma criação simplificada para índices, *uptades* rápida dos índices e uma herança de um controle de concorrência da *B-Tree* [KT, 2002]. Em relação ao R-Tree as operações de *updates*, não contribuem para a perda do desempenho.

Nesta secção feita uma breve descrição sobre os dados espaciais, incluindo o tipo de dados mais utilizado em SIG e as operações realizada sobre esses dados.

No seguimento foi analisada vários métodos de indexação espacial para indexar dados do tipo ponto com dando especial relevância aos que permitem optimizar operações topológicas, nomeadamente operações de vizinhança local.

Ainda foram apresentados os SGBS com extensão espacial e elaborado um quadro comparativo destacando algumas funcionalidades entre elas.

Por fim foi analisado algumas estratégias que permitem reduzir as operações de I/O e maximizar o desempenho no acesso aos dados.

Processamento de pesquisas

De maneira a aumentar o desempenho nas buscas, o Oracle Spatial utiliza o modelo de busca multifase. No primeiro filtro denominado primário como se ilustra na Figura 20 primeiramente os índices são utilizados como filtro de busca [KT, 2002]. Para realizar uma busca espacial, no qual se procura encontrar determinadas características que satisfaçam um determinado predicado, as geometrias candidatas a satisfazerem o predicado são relacionadas primeiramente através de índices espaciais. Nesta fase as aproximações nos índices (os MBRs armazenados na tabela espacial), são usados para identificar os conjuntos de candidatos que satisfazem as relações de topologia especificadas pela querie de pesquisa.

No filtro intermediário, as geometrias candidatas são comparadas utilizando aproximações interiores das geometrias de buscas e as candidatas. Algumas são seleccionadas e outras não dependendo do critério de busca.

A última fase de busca, denominado filtro secundário, é onde são realizados cálculos computacionais com as geometrias exactas. Esses cálculos são computacionalmente dispendiosos e determinam o resultado exacto que eventualmente são retornados ao utilizador. A Figura 15 ilustra as filtragens necessárias para avaliar uma querie espacial.

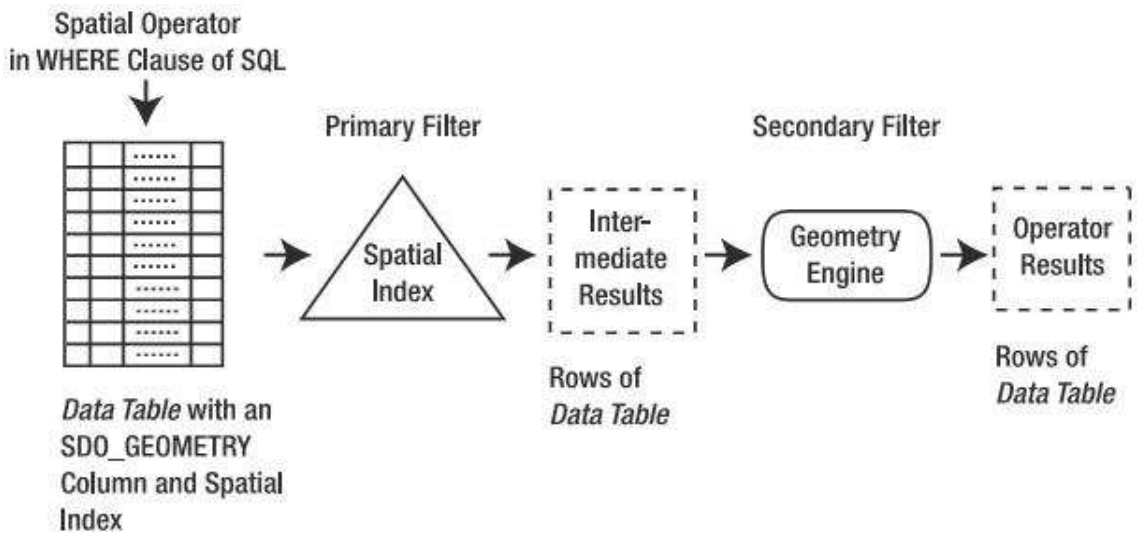

Figura 21: Processamento de pesquisas espaciais no Oracle, fonte [RAE, 2007].

Todos os processamentos citados anteriormente são transparentes para o utilizador; especificando apenas os operadores na cláusula *WHERE* num procedimento SQL, é invocado internamente o índice mais adequado (filtro primário) e o motor da execução da geometria representado na Figura 15 Geometry Filter (filtro secundário), para identificar o conjunto correcto de colunas.

A última fase de busca, denominado filtro secundário, é onde são realizados cálculos computacionais com as geometrias exactas. Esses cálculos são computacionalmente dispendiosos

Operadores Espaciais no Oracle

A componente espacial do Oracle já disponibiliza um conjunto de operadores espaciais que permite a manipulação de operações de vizinhança local. Podem ser aplicados em diferentes casos como se detalha em baixo:

- Por exemplo dado um ponto *p*, encontrar todos os pontos *x* num conjunto *P*, de dados tal que a distância de $d(x, p) \leq r$, onde r é o raio de um círculo centrado em *p*. Para o cálculo desta operação é disponibilizado o operador *SDO_WITHIN_DISTANCE,* ou simplesmente o operador de distância *within.*
- Para encontrar os vizinhos mais próximos de uma pesquisa é usado o operador *SDO_NN* ou então o operador *nearest-neighbor*.
- Em outros casos interessa saber os vizinhos que cruzam ou relacionam com uma *querie* de pesquisa. O primeiro operador para realizar esta operação é o *SDO_RELATE,* no entanto existem outras variantes para determinar tipos específicos de relações. Caso for utilizado apenas as aproximações dos índices, é possível usar uma variante do operador *SDO_RELATE,* chamado de *SDO_FILTER.*

Para mais informações sobre os operadores espaciais, implementados no Oracle Spatial consultar [RAE07].

Funcionamento da QuadTree como filtro primário

A QuadTree necessita que o utilizador defina um *tilling level,* (números de subdivisões recursivas do espaço). Cada geometria é então aproximada por um conjunto mínimo de quadrantes. Durante a subdivisão do espaço, conhecido também como *tesselation,* os quadrantes são divididos entre quadrantes interiores e de fronteiras, dependendo se todos estão no interior da geometria ou não. Todos os quadrantes que envolvem a geometria são então introduzidos numa tabela de índices espaciais, onde cada linha guarda diferentes códigos de quadrante interior/exterior e o *rowID* da geometria (linha da geometria na tabela). Uma B-Tree é então criada contendo o código do quadrante e o *rowID* da geometria.

Quando se realiza uma busca que envolve uma geometria de busca como parâmetro, a geometria ocorre uma *tesselation* para a mesma. Usando cada quadrante da geometria de busca e o *rowID* chegamos a todos as geometrias que possuem quadrantes com a mesma posição dos quadrantes de geometria de busca.

Quanto maior o *tilling,* mais precisas são as aproximações das geometrias, consequentemente mais refinadas são as buscas. Encontrar o *tilling level,* correcto é crucial, dado que os quadrantes podem ser pequeno demais ou grande demais [KT, 2002]. Quanto maior o *tilling level,* mais selectivo é o filtro primário, consequentemente menos geometrias são submetidas ao filtro secundário. Existe uma função da Oracle que estima o *tilling level* correcto para as geometrias da Base de dados; a *SDO_ESTIMATE_TILING_LEVEL ();*

Funcionamento da R-Tree como filtro primário

Na Oracle Spatial, a R-Tree mantém a sua estrutura lógica de árvore e são implementadas como uma tabela onde cada nó da R-Tree corresponde a uma linha em uma tabela e um apontador filho na R-Tree corresponde a um rowID, na mesma tabela. A raiz da árvore está presente em metadados o que permite uma navegação da raiz a folha da mesma árvore. A R-Tree mantém a sua estrutura lógica da árvore, porém estão implementados como uma tabela onde cada nó da árvore corresponde a uma linha da tabela. A raiz da tabela está na tabela de metadados o que permite uma navegação da raiz aos nós folhas da árvore. A Figura 22 ilustra um exemplo do índice espacial R-Tree. No lado esquerdo é representado os pontos e os MBRs dos nós, enquanto no lado direito é representado a organização numa estrutura R-Tree.

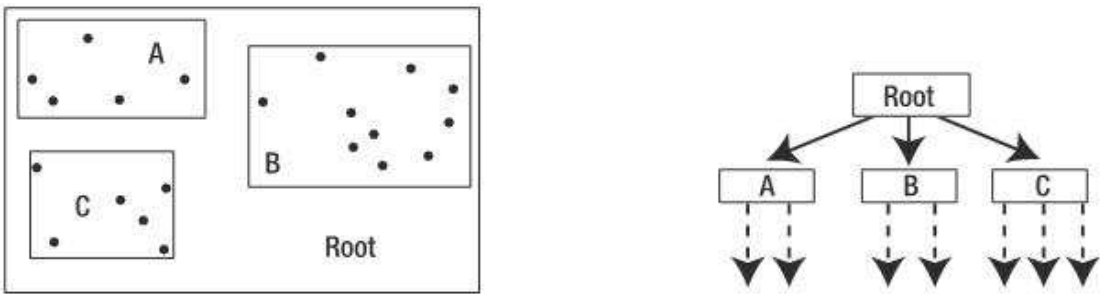

Figura 22: R-Tree Spatial índex no Oracle Spatial, fonte [RAE, 2007]

Os pontos representam as coordenadas dos dados espaciais, armazenados como uma geometria do tipo ponto (*SDO_GEOMETRY*), representando a coluna *pontos* na tabela FMA. Para cada geometria especificada na criação da tabela, a R-Tree calcula o MBR que engloba a geometria e calcula uma hierarquia de MBRs.

No exemplo apresentado em cima, Figura 17 a localização dos pontos estão estruturados em três nós. Cada nó é associado com um MBR que engloba a localização dos dados na sub-árvore. Estes nós são armazenados numa única raiz, permitindo a R-Tree construir uma árvore hierarquia usando os MBRs do tipo *SDO_GEOMETRY* na tabela. De seguida é usado as hierarquias dos MBRs para orientar as pesquisas para os nós pretendidos e por fim para as colunas das tabelas. Na é apresentado como os índices R-Tree são armazenados no Oracle.

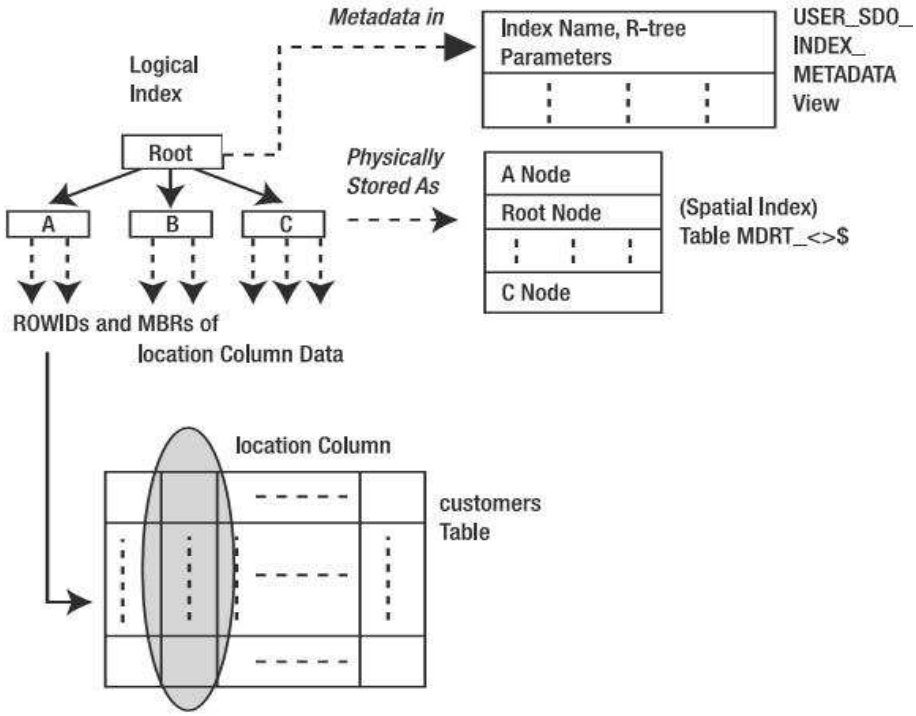

Figura 23: Armazenamento do índice espacial R-Tree, fonte [RAE, 2007]

A estrutura lógica é armazenada como uma tabela no Oracle começando com o prefixo *MDRT*. A metadata para os índices é armazenada na *view USER_SDO_INDEX_METADATA.* Esta *view* armazena os nomes dos índices espaciais como *SDO_INDEX_NAME;* as tabelas armazenam os índices como *SDO_INDEX_TABL;* a raiz do *ROWID* para a R-Tree; informações sobre a quantidade de entradas que um nó é capaz de suportar denominado por *fanout* ou *branching factor*, e entre outros parâmetros.

ANEXO IV: Resultados Globais dos Testes

4.1 FMA Original

Teste 1: 2500 pontos:

A ler... 2500Pontos...Correcto
A criar estrutura de dados... Ok
Dados Carregados com sucesso
A iniciar Filtro:
A verificar parametros...Ok...

A iniciar iteração 1 de 5
Duração Total(seg): 0.608
Das quais:
Em distancias e Min(seg): 0.607
Em descriminação dos pontos(seg): 0
A iniciar iteração 2 de 5
Duração Total(seg): 0.576
Das quais:
Em distancias e Min(seg): 0.576
Em descriminação dos pontos(seg): 0
A iniciar iteração 3 de 5
Duração Total(seg): 0.62
Das quais:
Em distancias e Min(seg): 0.62
Em descriminação dos pontos(seg): 0
A iniciar iteração 4 de 5
Duração Total(seg): 0.715
Das quais:
Em distancias e Min(seg): 0.715
Em descriminação dos pontos(seg): 0
A iniciar iteração 5 de 5
Duração Total(seg): 0.886
Das quais:
Em distancias e Min(seg): 0.886
Em descriminação dos pontos(seg): 0
Tempo total de filtragem(seg): 3.451

Teste 2: 10000

A ler... 10000 Pontos...Correcto
A criar estrutura de dados... Ok
Dados Carregados com sucesso
A iniciar Filtro:
A verificar parametros...Ok...

A iniciar iteração 1 de 5
Duração Total(seg): 8.843
Das quais:
Em distancias e Min(seg): 8.843
Em descriminação dos pontos(seg): 0

A iniciar iteração 2 de 5
Duração Total(seg): 8.995
Das quais:
Em distancias e Min(seg): 8.995
Em descriminação dos pontos(seg): 0
A iniciar iteração 3 de 5
Duração Total(seg): 9.418
Das quais:
Em distancias e Min(seg): 9.418
Em descriminação dos pontos(seg): 0
A iniciar iteração 4 de 5
Duração Total(seg): 10.274
Das quais:
Em distancias e Min(seg): 10.273
Em descriminação dos pontos(seg): 0.001
A iniciar iteração 5 de 5
Duração Total(seg): 11.828
Das quais:
Em distancias e Min(seg): 11.828
Em descriminação dos pontos(seg): 0
Tempo total de filtragem(seg): 49.398

Teste 3 : 25000

A ler... 25000Pontos...Correcto
A criar estrutura de dados... Ok
Dados Carregados com sucesso
A iniciar Filtro:
A verificar parametros...Ok...

A iniciar iteração 1 de 5
Duração Total(seg): 54.819
Das quais:
Em distancias e Min(seg): 54.818
Em descriminação dos pontos(seg): 0.001
A iniciar iteração 2 de 5
Duração Total(seg): 55.835
Das quais:
Em distancias e Min(seg): 55.834
Em descriminação dos pontos(seg): 0.001
A iniciar iteração 3 de 5
Duração Total(seg): 57.713
Das quais:
Em distancias e Min(seg): 57.712
Em descriminação dos pontos(seg): 0.001
A iniciar iteração 4 de 5
Duração Total(seg): 61.836
Das quais:
Em distancias e Min(seg): 61.835
Em descriminação dos pontos(seg): 0.001
A iniciar iteração 5 de 5
Duração Total(seg): 69.641
Das quais:
Em distancias e Min(seg): 69.64
Em descriminação dos pontos(seg): 0.001
Tempo total de filtragem(seg): 299.884

Teste 5:

A ler... 50000 Pontos...Correcto
A criar estrutura de dados... Ok
Dados Carregados com sucesso
A iniciar Filtro:
A verificar parametros...Ok...

A iniciar iteração 1 de 5
Duração Total(seg): 232.546
Das quais:
Em distancias e Min(seg): 232.544
Em descriminação dos pontos(seg): 0.002
A iniciar iteração 2 de 5
Duração Total(seg): 235.433
Das quais:
Em distancias e Min(seg): 235.431
Em descriminação dos pontos(seg): 0.002
A iniciar iteração 3 de 5
Duração Total(seg): 242.725
Das quais:
Em distancias e Min(seg): 242.723
Em descriminação dos pontos(seg): 0.002
A iniciar iteração 4 de 5
Duração Total(seg): 257.374
Das quais:
Em distancias e Min(seg): 257.372
Em descriminação dos pontos(seg): 0.002
A iniciar iteração 5 de 5
Duração Total(seg): 285.981
Das quais:
Em distancias e Min(seg): 285.979
Em descriminação dos pontos(seg): 0.002
Tempo total de filtragem(seg): 1254.1

Teste 6: 200000

A ler... 200.000 Pontos...Correcto
A criar estrutura de dados... Ok
Dados Carregados com sucesso
A iniciar Filtro:
A verificar parametros...Ok...

A iniciar iteração **1 de 4**
Duração Total(min): 92.209 : 1,53 horas
Das quais:
Em distancias e Min(seg): 5532.5244
Em descriminação dos pontos(seg): 0.0156

A iniciar iteração **2 de 4**
Duração Total(min): 93.27838: 1,55 horas
Das quais:
Em distancias e Min(seg): 5596.6872
Em descriminação dos pontos(seg): 0.0156

A iniciar iteração **3 de 4**
Duração Total(min): 95.67272 : 1,59 horas
Das quais:
Em distancias e Min(seg): 5740.3476
Em descriminação dos pontos(seg): 0.0156

A iniciar iteração **4 de 4**
Duração Total(min): 101.37019:1,69 horas
Das quais:
Em distancias e Min(seg): 6082.1958
Em descriminação dos pontos (seg): 0.0156

Tempo total de filtragem (min):382.53055: 6,32 horas

4.2 FMA Original Optimizado Memória

A ler... 2500Pontos...Correcto

A criar estrutura de dados... Ok

Dados Carregados com sucesso

Ok

A iniciar Filtro:

A verificar paramentos... Ok...

A organizar estrutura de dados espaciais...

Este processo poderá demorar dependendo do tamanho de dados...

Dados espaciais organizados com sucesso

Tempo de execução: 4.156383

Media por ponto: 0.0016625532

A iniciar iteração 1 de 4

Duração Total(seg): 0.2968845

Das quais:

Em distancias e Min(seg): 0

Em descriminação dos pontos(seg): 0

A iniciar iteração 2 de 4

Duração Total(seg): 0.343761

Das quais:

Em distancias e Min(seg): 0

Em descriminação dos pontos(seg): 0

A iniciar iteração 3 de 4

Duração Total(seg): 0.5156415

Das quais:

Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0
A iniciar iteração 4 de 4
Duração Total(seg): 1.0781595
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0
Tempo total de filtragem(seg): 6.3908295

A ler... 25000Pontos...Correcto
A criar estrutura de dados... Ok
Dados Carregados com sucesso
Ok
A iniciar Filtro:
A verificar parametros...Ok...
A organizar estrutura de dados espaciais...
Este processo poderá demorar dependendo do tamanho de dados...
Dados espaciais organizados com sucesso
Tempo de execução: 57.6737205
Media por ponto: 0.00230694882
A iniciar iteração 1 de 4
Duração Total(seg): 3.156351
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0
A iniciar iteração 2 de 4
Duração Total(seg): 3.6094905
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0
A iniciar iteração 3 de 4
Duração Total(seg): 5.250168
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0
A iniciar iteração 4 de 4
Duração Total (seg): 11.625372

Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0
Tempo total de filtragem (seg): 81.315102

A ler... 50000Pontos...Correcto
A criar estrutura de dados... Ok
Dados Carregados com sucesso
Ok
A iniciar Filtro:
A verificar parametros...Ok...
A organizar estrutura de dados espaciais...
Este processo poderá demorar dependendo do tamanho de dados...
Dados espaciais organizados com sucesso
Tempo de execução: 178.9901025
Media por ponto: 0.00357980205
A iniciar iteração 1 de 4
Duração Total(seg): 6.8908455
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0
A iniciar iteração 2 de 4
Duração Total(seg): 7.343985
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0
A iniciar iteração 3 de 4
Duração Total(seg): 10.719093
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0
A iniciar iteração 4 de 4
Duração Total(seg): 24.37578
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0
Tempo total de filtragem (seg): 228.319806

A ler... 200000Pontos...Correcto
A criar estrutura de dados... Ok
Dados Carregados com sucesso
Ok
A iniciar Filtro:
A verificar parametros...Ok...
A organizar estrutura de dados espaciais...
Este processo poderá demorar dependendo do tamanho de dados...
Dados espaciais organizados com sucesso
Tempo de execução: 2136.1625325
Media por ponto: 0.0106808126625
A iniciar iteração 1 de 4
Duração Total(seg): 35.3605065
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0
A iniciar iteração 2 de 4
Duração Total(seg): 38.2668495
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0
A iniciar iteração 3 de 4
Duração Total(seg): 55.970541
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0
A iniciar iteração 4 de 4
Duração Total(seg): 125.4883905
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0
Tempo total de filtragem(seg): 2391.24882

4.3 FMA Original Optimizado com acesso aos discos

R-TREE

A ligar à BD... Ok
A carregar dados... Ok
Total de pontos... 2500A ler parametros... Ok
A organizar estrutura de dados espaciais...
Este processo poderá demorar dependendo do tamanho de dados...
Dados espaciais organizados com sucesso
Tempo de execução: 126.1600733
Media por ponto: 0.05046402932

A iniciar iteração 1 de 4
Duração Total(seg): 0.31251
Duração Total(seg): 0.0052085
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 2 de 4
Duração Total(seg): 0.343761
Duração Total(seg): 0.00572935
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 3 de 4
Duração Total(seg): 0.500016
Duração Total(seg): 0.0083336
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 4 de 4
Duração Total(seg): 1.0781595
Duração Total(seg): 0.017969325
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0
Tempo total de filtragem(seg): 128.4101453
Tempo total de filtragem(seg): 128.4101453
Tempo total de filtragem(Minuntos): 2.14016908833333

A ligar à BD... Ok
A carregar dados... Ok
Total de pontos... 25000A ler parametros... Ok
A organizar estrutura de dados espaciais...
Este processo poderá demorar dependendo do tamanho de dados...
Dados espaciais organizados com sucesso
Tempo de execução: 1241.0440208
Media por ponto: 0.049641760832

A iniciar iteração 1 de 4
Duração Total(seg): 3.1719359
Duração Total(seg): 0.0528655983333333

Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 2 de 4
Duração Total(seg): 3.6250696
Duração Total(seg): 0.0604178266666667
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 3 de 4
Duração Total(seg): 5.3594779
Duração Total(seg): 0.0893246316666667
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 4 de 4
Duração Total(seg): 11.9221039
Duração Total(seg): 0.198701731666667
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0
Tempo total de filtragem(seg): 1265.169484
Tempo total de filtragem(seg): 1265.169484
Tempo total de filtragem(Minuntos): 21.0861580666667

Double click para fechar relatório

A ligar à BD... Ok
A carregar dados... Ok
Total de pontos... 49999A ler parametros... Ok
A organizar estrutura de dados espaciais...
Este processo poderá demorar dependendo do tamanho de dados...
Dados espaciais organizados com sucesso
Tempo de execução: 2555.203782
Media por ponto: 0.0511050977419548

A iniciar iteração 1 de 4
Duração Total(seg): 6.2970765
Duração Total(seg): 0.104951275
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 2 de 4
Duração Total(seg): 7.093977
Duração Total(seg): 0.11823295
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 3 de 4
Duração Total(seg): 10.62534

Duração Total(seg): 0.177089
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 4 de 4
Duração Total(seg): 24.125772
Duração Total(seg): 0.4020962
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0
Tempo total de filtragem(seg): 2603.361573
Tempo total de filtragem(seg): 2603.361573
Tempo total de filtragem(Minuntos): 43.38935955

A ligar à BD... Ok
A carregar dados... Ok
Total de pontos... 200001A ler parametros... Ok
A organizar estrutura de dados espaciais...
Este processo poderá demorar dependendo do tamanho de dados...
Dados espaciais organizados com sucesso
Tempo de execução: 12423.47847
Media por ponto: 0.0621170817645912

A iniciar iteração 1 de 4
Duração Total(seg): 48.689058
Duração Total(seg): 0.8114843
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 2 de 4
Duração Total(seg): 51.8922855
Duração Total(seg): 0.864871425
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 3 de 4
Duração Total(seg): 56.2361745
Duração Total(seg): 0.937269575
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 4 de 4
Duração Total(seg): 126.2071635
Duração Total(seg): 2.103452725
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0
Tempo total de filtragem(seg): 12706.5031515
Tempo total de filtragem(seg): 12706.5031515
Tempo total de filtragem(Minuntos): 211.775052525

QUADTREE

A ligar à BD... Ok
A carregar dados... Ok
Total de pontos... 2500A ler parametros... Ok
A organizar estrutura de dados espaciais...
Este processo poderá demorar dependendo do tamanho de dados...
Dados espaciais organizados com sucesso
Tempo de execução: 117.1756245
Media por ponto: 0.0468702498

A iniciar iteração 1 de 4
Duração Total(seg): 0.2968845
Duração Total(seg): 0.004948075
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 2 de 4
Duração Total(seg): 0.3281355
Duração Total(seg): 0.005468925
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 3 de 4
Duração Total(seg): 0.500016
Duração Total(seg): 0.0083336
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 4 de 4
Duração Total(seg): 1.0781595
Duração Total(seg): 0.017969325
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0
Tempo total de filtragem(seg): 119.410071
Tempo total de filtragem(seg): 119.410071
Tempo total de filtragem(Minuntos): 1.99016785

Double click para fechar relatório

A ligar à BD... Ok
A carregar dados... Ok
Total de pontos... 25000A ler parametros... Ok
A organizar estrutura de dados espaciais...
Este processo poderá demorar dependendo do tamanho de dados...
Dados espaciais organizados com sucesso
Tempo de execução: 1278.6853815
Media por ponto: 0.05114741526

A iniciar iteração 1 de 4
Duração Total(seg): 3.1719765

Duração Total(seg): 0.052866275
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 2 de 4
Duração Total(seg): 3.625116
Duração Total(seg): 0.0604186
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 3 de 4
Duração Total(seg): 5.3282955
Duração Total(seg): 0.088804925
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 4 de 4
Duração Total(seg): 11.8285035
Duração Total(seg): 0.197141725
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0
Tempo total de filtragem(seg): 1302.639273
Tempo total de filtragem(seg): 1302.639273
Tempo total de filtragem(Minuntos): 21.71065455

A ligar à BD... Ok
A carregar dados... Ok
Total de pontos... 49999A ler parametros... Ok
A organizar estrutura de dados espaciais...
Este processo poderá demorar dependendo do tamanho de dados...
Dados espaciais organizados com sucesso
Tempo de execução: 2557.8474735
Media por ponto: 0.0511579726294526

A iniciar iteração 1 de 4
Duração Total(seg): 6.812718
Duração Total(seg): 0.1135453
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 2 de 4
Duração Total(seg): 7.875252
Duração Total(seg): 0.1312542
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos (seg): 0

A iniciar iteração 3 de 4
Duração Total (seg): 11.3909895
Duração Total (seg): 0.189849825

Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 4 de 4
Duração Total(seg): 24.875796
Duração Total(seg): 0.4145966
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0
Tempo total de filtragem(seg): 2608.802229
Tempo total de filtragem(seg): 2608.802229
Tempo total de filtragem(Minuntos): 43.48003715

A ligar à BD... Ok
A carregar dados... Ok
Total de pontos... 200001A ler parametros... Ok
A organizar estrutura de dados espaciais...
Este processo poderá demorar dependendo do tamanho de dados...
Dados espaciais organizados com sucesso
Tempo de execução: 12249.7330125
Media por ponto: 0.0612483588207059

A iniciar iteração 1 de 4
Duração Total(seg): 49.6422135
Duração Total(seg): 0.827370225
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 2 de 4
Duração Total(seg): 53.1267
Duração Total(seg): 0.885445
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 3 de 4
Duração Total(seg): 56.689314
Duração Total(seg): 0.9448219
Das quais:
Em distancias e Min(seg): 0
Em descriminação dos pontos(seg): 0

A iniciar iteração 4 de 4
Duração Total (seg): 127.2696975
Duração Total (seg): 2.121161625
Das quais:
Em distancias e Min (seg): 0
Em descriminação dos pontos (seg): 0
Tempo total de filtragem (seg): 12536.8046985
Tempo total de filtragem (Minuntos): 208.946744975