UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

Relatórios Técnicos
do Departamento de Informática Aplicada
da UNIRIO
n° 0003/2011

Cálculo da Similaridade de Polígonos Utilizando Assinaturas Raster

Léo Carvalho Ramos Antunes
Leonardo Guerreiro Azevedo

Departamento de Informática Aplicada

UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
Av. Pasteur, 458, Urca - CEP 22290-240
RIO DE JANEIRO – BRASIL

Relatórios Técnicos do DIA/UNIRIO, No. 0003/2011
Editor: Prof. Sean W. M. Siqueira Janeiro, 2011

Cálculo da Similaridade de Polígonos Utilizando Assinaturas Raster

Léo Carvalho Ramos Antunes, Leonardo Guerreiro Azevedo

Depto de Informática Aplicada – Universidade Federal do Estado do Rio de Janeiro (UNIRIO)

{leo.antunes, azevedo}@uniriotec.br

Abstract. The similarity concept is fundamental for learning, knowledge and thought. Many science areas develop their own notions of similarity. This work presents an algorithm to compute similarity between polygons through their four-color raster signatures (4CRS). Required implementations for this algorithm were performed in SECONDO, an extensible DBMS platform. Experimental tests were executed in order to evaluate algorithm precision with respect to compute similarity through real polygons.

Keywords: Spatial database, Approximate Query Processing, Similarity between spatial objects, SECONDO.

Resumo. O conceito de similaridade é fundamental para a aprendizagem, conhecimento e reflexão. Muitas áreas da ciência têm suas próprias noções de similaridade. Este trabalho apresenta um algoritmo para cálculo da similaridade entre polígonos a partir de suas assinaturas raster de quatro cores (4CRS). As implementações necessárias para este algoritmo foram realizadas no SECONDO, um banco de dados extensível. Testes experimentais foram realizados a fim de avaliar a precisão do algoritmo em relação ao cálculo da similaridade utilizando os próprios polígonos.

Palavras-chave: Banco de dados espaciais, Processamento aproximado de consultas, Similaridade entre objetos espaciais, SECONDO.

Sumário

1 Introdução

O objetivo deste capítulo é contextualizar o assunto abordado neste trabalho, bem como apresentar sua motivação e a estrutura dos capítulos

1.1 Motivação

Segundo Quine [1969] e Cakmakov e Celakoska [2004], o conceito de similaridade é fundamental para a aprendizagem, conhecimento e reflexão. Uma métrica de similaridade fornece uma medida de semelhança entre pares de coisas que permite identificar a que classes elas pertencem. O processo de classificar objetos é uma característica fundamental da maioria das atividades humanas, e a idéia de que pessoas classificam objetos juntos que julgam ser similares é intuitiva e popular entre várias disciplinas [Holt, 2003]. [Tversky, 1977] descreve o conceito de similaridade como um princípio de organização pelo qual pessoas classificam objetos, conceitos de forma, e fazem generalizações.

Muitos autores concordam que diferentes áreas da ciência desenvolvem suas próprias noções para similaridade. [Goodman, 1972] afirma que não há nada como similaridade global, que pode ser medida, mas é sempre necessário dizer sob quais aspectos duas coisas são similares. Avaliações de similaridade vão, portanto, se tornando crucialmente dependentes do contexto em que elas ocorrem [Cakmakov e Celakoska, 2004].

Áreas de aplicação para uso da similaridade incluem: bancos de dados de imagens médicas, reconhecimento de gestos/movimentos humanos, sistemas de informações geológicas, comércio eletrônico, proteção de direitos autorais, design gráfico, arte criativa etc. [Sako e Fujimura, 2000].

Várias teorias filosóficas e psicológicas de similaridade têm sido aplicadas na ciência da informação. Algumas dessas aplicações relacionadas com a similaridade são: indexação, *sub-setting*, recuperação, correspondência, *ranking*, espaço de solução, *clustering*, árvores, categorização, igualdade e equivalência. Similaridade espacial pode ser vista como um subconjunto de similaridade na qual as entidades que estão sendo comparadas possuem componentes espaciais [Holt, 2003].

Este trabalho trata da similaridade entre objetos espaciais. Dados espaciais consistem de pontos, linhas, regiões, retângulos, superfícies e volumes [Samet, 1990]. São exemplos de dados espaciais: cidades, rios, estradas, países, estados, áreas de plantio, cadeias de montanhas etc. Freqüentemente, atributos espaciais aparecem associados com atributos não espaciais. Exemplos de dados não espaciais são: nomes de estradas, endereços, números de telefone, nomes de cidades etc. [Azevedo *et al.*, 2006].

Sistemas Gerenciadores de Banco de Dados (SGBDs ou Database Management Systems - DBMS) tradicionais são voltados para o uso de dados escalares (unidimensionais) e, portanto, não são indicados para apoiar aplicações que utilizam dados multidimensionais, podendo ser espaciais ou espaço-temporais [Theodoridis *et al.*, 1998].

Segundo [Güting, 1994], Sistemas Gerenciadores de Banco de Dados Espaciais (SGBDE ou Spatial Database Management Systems - SDBMS) provêem a tecnologia de banco de dados fundamental para Sistemas de Informações Geográficas (SIG ou Geographic Information Systems - GIS) e outras aplicações. Um SGBDE possui as seguintes características:

- É um sistema de banco de dados.
- Oferece tipos de dados espaciais (Spatial Datatypes – SDT's) no seu modelo de dados e na sua linguagem de consulta.
- Suporta tipos de dados espaciais na sua implementação provendo, pelo menos, indexação espacial e algoritmos eficientes para junções espaciais.

Existem numerosas aplicações na área de banco de dados espaciais: supervisão de tráfego, controle aéreo, previsão do tempo, planejamento urbano, otimização de rota, cartografia, agricultura, administração de recursos naturais, monitoramento costeiro, fogo e controle de epidemias [Aronoff, 1989; Tao *et al.*, 2003].

Uma questão principal na área de banco de dados é o processamento eficiente de consultas para que os usuários não tenham que esperar um longo tempo para receber uma resposta. Entretanto, existem muitos casos onde não é fácil alcançar este requisito. Além disto, uma resposta rápida pode ser mais importante para o usuário do que receber uma resposta exata. Em outras palavras, a precisão da consulta pode ser diminuída e uma resposta aproximada retornada, desde que o processamento seja rápido e a precisão aceitável.

Uma das formas de executar processamento aproximado de consultas é utilizar aproximações dos dados ao invés de utilizar dados reais. [Zimbrão e Souza, 1998] propuseram a Assinatura Raster de Quatro Cores (4CRS – Four-Colour Raster Signature) para aproximar objetos espaciais descritos como polígonos.

1.2 Objetivo

Este trabalho propõe um algoritmo para calcular a similaridade entre polígonos através de suas assinaturas 4CRS. Esta proposta considera comparar objetos de acordo com a sua posição no espaço, ou seja, onde a posição absoluta é relevante. Por exemplo, na Figura 1 e na Figura 2 temos os mesmos polígonos onde na primeira existe sobreposição entre eles e na segunda não há sobreposição. Para o primeiro caso, a partir das assinaturas 4CRS dos polígonos apresentados na Figura 1, é retornada um valor maior do que zero que indica o quanto os polígonos são similares. No segundo caso, será retornado que a semelhança entre os polígonos é igual a zero. Uma evolução do algoritmo aqui proposto seria deslocar objetos para posições no espaço a fim de comparar a forma dos objetos, não considerando sua posição relativa.

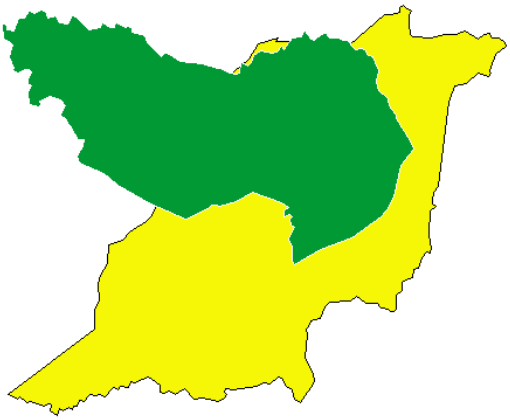

Figura 1 – Exemplos de polígonos com similaridade maior do que zero

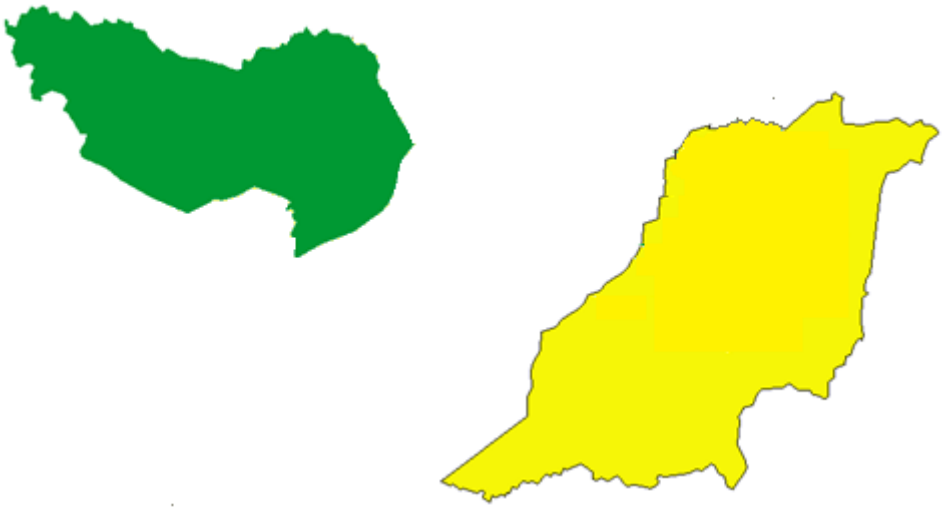

Figura 2 - Exemplos de polígonos com similaridade igual a zero

O algoritmo retorna um valor entre 0 e 1, indicando a semelhança aproximada dos objetos. O algoritmo para computar a similaridade utiliza outros algoritmos já propostos na literatura (o algoritmo para calcular a área aproximada de polígonos, proposto por [Azevedo *et al.*, 2004] e o algoritmo para calcular a área aproximada de interseção entre polígonos, proposto por [Azevedo *et al.*, 2005]). Além destes algoritmos, também é necessário calcular a assinatura correspondente à união de duas assinaturas 4CRS. Este algoritmo foi proposto neste trabalho e corresponde a uma contribuição do mesmo. Outra contribuição é a implementação destes algoritmos no SECONDO. O SECONDO é um Sistema Gerenciador de Banco de Dados extensível que suporta principalmente tipos de dados não-convencionais como, por exemplo, dados espaciais [Güting *et al.*, 2005]. A fim de avaliar a precisão do algoritmo, foram realizados testes experimentais utilizando conjuntos de dados compostos por polígonos de municípios da região norte do Brasil.

1.3 Estrutura do trabalho

Este trabalho está dividido da seguinte forma. A Seção 1 corresponde a presente introdução e apresenta a motivação e contextualiza o assunto deste trabalho. A Se-

ção 2 apresenta os principais conceitos do trabalho, tais como: Processamento Aproximado de Consultas, Assinatura Raster de Quatro Cores (4CRS) e Processamento Aproximado de Consultas utilizando a 4CRS. A Seção 3 apresenta a proposta de algoritmo para similaridade entre polígonos. A Seção 4 apresenta a implementação do algoritmo no SECONDO. A Seção 5 apresenta os testes experimentais realizados, bem como as análises dos resultados obtidos. A Seção 6 apresenta a conclusão do trabalho. Em seguida, são apresentadas as referências bibliográficas utilizadas no trabalho e os apêndices que detalham: as configurações necessárias para incluir uma álgebra no SECONDO (Apêndice 1), como implementar um novo operador em uma álgebra (Apêndice 2) e os códigos-fonte dos algoritmos implementados neste trabalho (Apêndice 3).

2 Principais conceitos

O objetivo deste capítulo é apresentar os conceitos básicos necessários sobre Processamento Aproximado de Consultas, Assinatura Raster de Quatro Cores (4CRS) e Processamento Aproximado de Consultas utilizando a 4CRS, utilizados neste trabalho.

2.1 Processamento Aproximado de Consultas

O processamento aproximado de consultas surgiu como uma alternativa para o processamento de consultas em ambientes nos quais o fornecimento de uma resposta exata pode demandar muito tempo. O objetivo é fornecer uma resposta estimada em ordem de magnitude de tempo menor do que o tempo necessário para computar uma resposta exata, evitando ou minimizando o número de acessos a disco ao banco de dados [Gibbons *et al*. 1997].

Alguns exemplos de aplicações que podem utilizar processamento aproximado de consultas são apresentados a seguir:

- [Hellerstein *et al.* 1997] enfatizam que em Sistemas de Apoio à Decisão (SAD ou Decision Support Systems - DSS) o crescimento na competitividade dos negócios está requerendo uma indústria baseada na informação. Logo, tornam-se necessárias técnicas de apresentação dos dados acumulados para os tomadores de decisão em tempo hábil. Eles também propõem o uso de processamento aproximado de consultas durante uma sequência de consultas *drill-down* em mineração de dados *ad-hoc*, onde as consultas anteriores na sequência são usadas somente para determinar quais são as consultas de interesse. Papadias *et al*. [2001] propõem o processamento aproximado de consultas para OLAP (Online Analytical Processing) espacial.
- Gibbons *et al.* [1997] destacam também que uma resposta aproximada pode fornecer feedback sobre quão bem colocada está a consulta. Além disto, pode ser usada quando a consulta demanda respostas numéricas e a precisão da resposta exata não é necessária. Por exemplo, uma média total, ou porcentagem para o qual somente os primeiros dígitos de preci-

são interessam (por exemplo, os primeiros dígitos de um valor que está na escala de milhão – 1.000.001 é aproximadamente 1.000.000).

2.2 Assinatura Raster de Quatro Cores (4CRS)

A Assinatura Raster de Quatro Cores (Four-Colour Raster Signature - 4CRS) foi proposta por Zimbrao e Souza [1998]. A 4CRS armazena as principais características dos dados em uma representação aproximada e compacta que pode ser acessada e processada mais rapidamente do que os dados reais. A assinatura corresponde a uma grade de células onde cada célula armazena informação relevante do objeto utilizando poucos bits (Tabela 1). A escala da grade pode ser modificada a fim de obter uma representação mais compacta (menor escala) ou com maior precisão (escala mais alta). Um exemplo de assinatura 4CRS é apresentado na Figura 3.

Tabela 1 - Tipos de célula 4CRS

Valor	Tipo	Descrição
00	Vazio	A célula não tem interseção com o polígono.
01	Pouco	A célula tem uma interseção de 50% ou menos com o polígono.
10	Muito	A célula tem uma interseção de mais de 50% e menos de 100% com o polígono.
11	Cheio	A célula é totalmente ocupada pelo polígono.

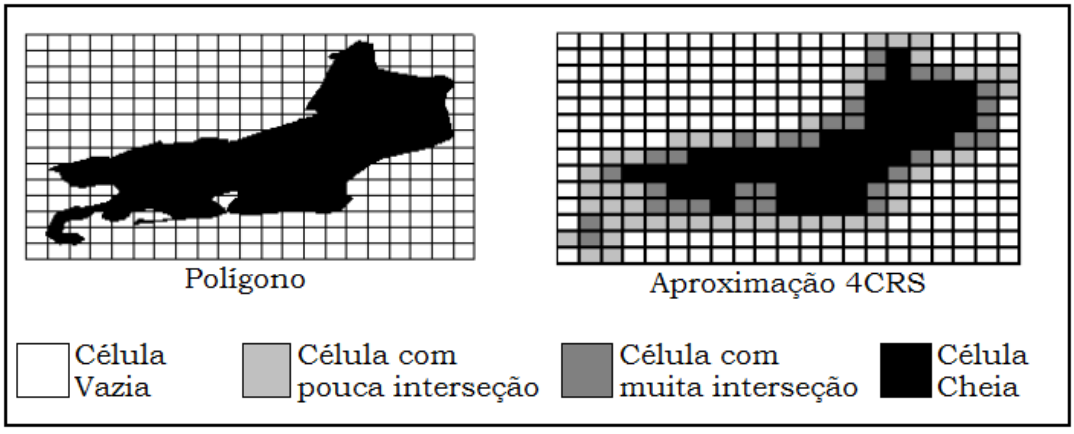

Figura 3 - Exemplo de uma assinatura 4CRS [Azevedo et al., 2004]

2.3 Processamento aproximado de consultas utilizando 4CRS

A assinatura 4CRS apresenta bons resultados para aproximação de polígonos em relação ao processamento exato de consultas, como apresentado por [Zimbrao e Souza, 1998] e [Monteiro *et al.*, 2004]. Isto motivou a avaliação do uso da 4CRS para processamento aproximado de consultas. Um conjunto de algoritmos foram pro-

postos e implementados por Azevedo *et al.* [2004, 2005 e 2006] e demonstraram bons resultados para o uso da 4CRS para processamento aproximado de consultas.

No processamento aproximado de consultas utilizando a 4CRS, ao invés de utilizar o objeto real, utiliza-se a assinatura 4CRS do objeto para realizar a consulta. Como resultado, uma resposta aproximada é retornada, juntamente com um intervalo de confiança. Como um exemplo, no cálculo da área aproximada de um polígono *p* [Azevedo *et al.*, 2004], o algoritmo calcula um valor aproximado de área *v*, e um intervalo de confiança *i*, considerando um percentual de confiança *c*. Portanto, a resposta final é que a área real é um valor entre *v-i* e *v+i*, com confiança igual a *c*.

3 Algoritmo para cálculo da similaridade raster

Neste trabalho, propomos um algoritmo para retornar a similaridade entre polígonos a partir de suas assinaturas 4CRS.

Uma função de similaridade, de forma intuitiva, retorna a similaridade de objetos em termos de tamanho, forma, e localização. Um exemplo de tal função para objetos espaciais que possuem área é calcular a similaridade de dois objetos a partir da razão entre a área de interseção e a área de união entre eles, como apresentado na Equação 1. Esta equação é uma medida intuitiva e conhecida como índice de Jaccard [Jaccard, 1912], como apontado por Hemert e Baldock [2007]. Yanchi *et al.* [2009] também apresentam esta equação com o intuito de definir uma medida de distância espacial.

$$S(o1, o2) = \frac{A_{\cap}(o1, o2)}{A_{\cup}(o1, o2)}$$

Onde:

- $o1$ e $o2$: *objetos espaciais que possuem área*
- $A_{\cap}$: *função que calcula a área de interseção entre dois objetos espaciais*
- $A_{\cup}$: *função que calcula a área de união entre dois objetos espaciais*

Equação 1 – Equação para calcular a similaridade entre polígonos

Este trabalho propõe o cálculo da similaridade de polígonos a partir da razão entre a área de interseção e a área de união de suas assinaturas 4CRS, como ilustrado na Equação 2.

$$S(r1, r2) = \frac{A_{\cap}(r1,r2)}{A_{\cup}(r1,r2)}$$

Onde:

- $r1$ e $r2$: *assinaturas raster de polígonos*
- $A_{\cap}$: *função que calcula a área de interseção entre dois polígonos a partir de suas assinaturas raster*
- $A_{\cup}$: *função que calcula a área de união entre dois polígonos a partir de suas assinaturas raster*

Equação 2 – Equação para calcular a similaridade entre polígonos a partir de suas assinaturas 4CRS

3.1 Algoritmos

Esta seção apresenta a proposta do algoritmo para cálculo de similaridade de polígonos a partir de suas assinaturas 4CRS, principal contribuição deste trabalho. Além disso, são apresentados os algoritmos utilizados pela proposta: algoritmo para calcular área aproximada, algoritmo para calcular área de interseção aproximada e algoritmo para calcular união de duas assinaturas 4CRS. Em particular, o algoritmo para calcular união de duas assinaturas 4CRS também consiste em uma contribuição deste trabalho.

3.1.1 Algoritmo para calcular a Similaridade

O algoritmo para calcular a similaridade (Figura 4) recebe como parâmetros duas assinaturas 4CRS de dois polígonos e retorna um valor no intervalo fechado [0,1], que indica o quanto eles são similares. Quanto mais este valor se aproxima de 1, mais similares são os objetos. Ele também retorna um intervalo de confiança que afirma com um percentual de certeza que o valor vai estar dentro deste intervalo. O algoritmo faz uso de outros três algoritmos: cálculo da área aproximada de um polígono (Figura 5), cálculo da área de interseção aproximada entre polígonos (Figura 7) e a união entre duas assinaturas 4CRS (Figura 9). Estes algoritmos são apresentados nas seções a seguir.

```
1. REAL similar(assinat4CRS1, assinat4CRS2)
2.        REAL    areaInter    =    areaIntersecaoAproximada(assinat4CRS1,
assinat4CRS2);
3.  SE(areaInter == 0) /* Não existe área de interseção */
4.   RETORNA 0;
5.  SENAO /* Existe área de interseção */
6.         ASSINAT4CRS   assinatUniao   =   uniaoAssinat4CRS(signat4CRS1,
signat4CRS2);
7.   REAL areaUniao = areaAproximada(assinatUniao);
8.  RETORNA areaInter / areaUniao;
```

Figura 4 – Algoritmo para cálculo da similaridade entre polígonos

3.1.2 Algoritmo para calcular a Área Aproximada

O algoritmo para calcular a área aproximada de um polígono a partir de sua assinatura 4CRS foi proposto por Azevedo *et al.* [2004]. O algoritmo soma a área estimada do polígono dentro de cada célula da grade. Para fazer isto, ele conta o número de células de cada tipo (Vazio, Pouco, Muito ou Cheio) na assinatura e multiplica pela média da área do polígono de acordo com cada tipo de célula. Por fim, ele multiplica esse resultado pela área total da célula, como apresentado na Equação 3.

$$A_{Aproximada} = (N_{Vazio} * P_{Vazio} + N_{Pouco} * P_{Pouco} + N_{Muito} * P_{Muito} + N_{Cheio} * P_{Cheio}) * A_{Celula}$$

Onde:

- $N_{tipoCelula}$ corresponde ao número de células daquele tipo presentes na assinatura. TipoCelula pode assumir os valores Vazio, Pouco, Muito ou Cheio;
- $P_{tipoCelula}$ corresponde à média da área do polígono dentro daquele tipo de célula;
- A_{celula} corresponde área total da célula.

Equação 3 – Equação para calcular a área aproximada

Células vazio não têm interseção com o polígono e células cheio são totalmente ocupadas pelo polígono, logo, a média da área de um polígono dentro de células vazio e cheio são, respectivamente, 0% e 100%. Porém, células pouco e muito necessitam de uma abordagem diferente. A área de um polígono dentro de uma célula pouco pode ser considerada como uma distribuição uniforme no intervalo aberto (0%, 50%) e dentro de uma célula muito (50%, 100%), como demonstrado por Azevedo *et al.* [2004]. Dessa forma, pode-se assumir que, em média, a área de um polígono dentro de uma célula pouco é de 25% e de uma célula muito é de 75%. A área do polígono dentro de cada tipo de célula é apresentada na Tabela 2. O algoritmo para calcular a área aproximada de um polígono é apresentado na Figura 5.

Tabela 2 – Área esperada do polígono dentro de cada tipo de célula

Tipo de célula	Área esperada
Vazio	0%
Pouco	25%
Muito	75%
Cheio	100%

```
1. REAL areaAproximada(assinat4CRS)
2.   numCelulasPouco = numCelulasMuito = numCelulasCheio = 0;
3.   areaCelula = assinat4CRS.tamanhoAresta * assinat4CRS.tamanhoAresta;
4.   PARA CADA celula EM assinat4CRS.celulas FACA
5.      SE (celula.tipo == POUCO)
6.         numCelulasPouco++;
7.      SENAO SE (celula.tipo == MUITO)
8.         numCelulasMuito++;
9.      SENAO SE (celula.tipo == CHEIO)
10.        numCelualasCheio++;
11.  RETORNA (numCelulasPouco * pesoCelulasPouco + numCelulasMuito *
       pesoCelulasMuito   +   numCelulasCheio   *   pesoCelulasCheio)   *
areaCelula;
```

Figura 5 - Algoritmo para calcular a área aproximada de um polígono

Azevedo [2005] apresenta um exemplo do uso do operador, o qual é descrito a seguir.

Considere o polígono e sua respectiva assinatura 4CRS apresentados na Figura 6. A assinatura contém a seguinte distribuição de células:

- Número de células *Vazio*: 210
- Número de células *Pouco*: 38
- Número de células *Muito*: 31
- Número de células *Cheio*: 61

Tendo o conhecimento que a área de uma célula da assinatura é igual a 262.144, para calcular a área do objeto a partir de sua assinatura 4CRS, devemos aplicar a Equação 3. Logo, vamos obter o seguinte resultado:

$$A_{Aproximada} = (210 * 0 + 38 * 0{,}25 + 31 * 0{,}75 + 61 * 1) * 262.144 = 24.576.000$$

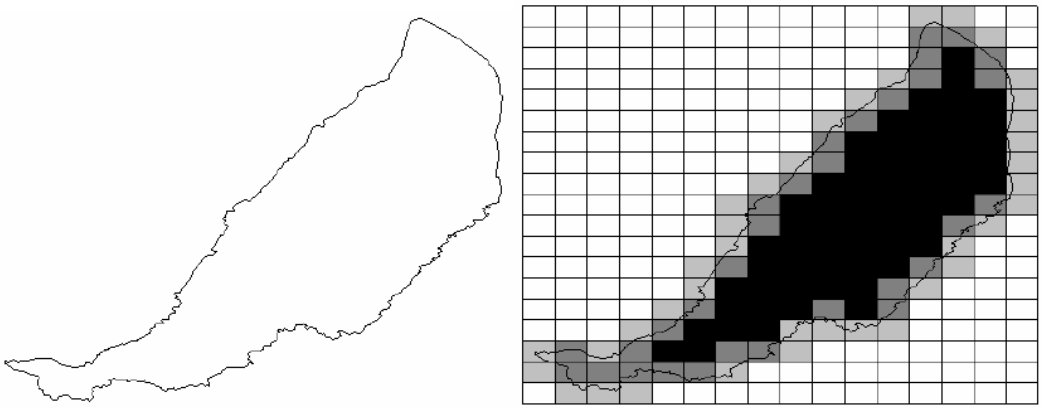

Figura 6 – Exemplo de polígono e sua assinatura 4CRS [Azevedo, 2005]

3.1.3 Algoritmo para calcular a Área de Interseção Aproximada

O algoritmo para calcular a área de interseção aproximada entre dois polígonos a partir de suas assinaturas 4CRS foi proposto por Azevedo *et al.* [2005]. Ele computa a soma das áreas estimadas das células das assinaturas que se sobrepõem, e multi-

plica o resultado pela área da célula. Como existem quatro tipos diferentes de células (Vazio, Pouco, Muito ou Cheio), as possibilidades de sobreposição são dezesseis. A área esperada de sobreposição em todos os casos é apresentada na Tabela 3. Maiores detalhes sobre os cálculos utilizados para definir estes valores são apresentados por Azevedo [2005].

Tabela 3 – Área esperada de sobreposição de tipos de células

Tipos de células	Vazio	Pouco	Muito	Cheio
Vazio	0	0	0	0
Pouco	0	0,0625	0,1875	0,25
Muito	0	0,1875	0,5625	0,75
Cheio	0	0,25	0,75	1

O algoritmo emprega uma matriz para armazenar as áreas esperadas. Apenas são consideradas no cálculo as células que estão contidas no MBR de interseção. Se as assinaturas possuírem tamanhos de célula distintos, para que possam ser comparadas, deve-se realizar uma mudança de escala da assinatura que possuir o menor tamanho de célula para o tamanho maior, pelos motivos descritos na Seção 2.2.1. Maiores detalhes para o entendimento destes cálculos são apresentados por Azevedo *et al.* [2005]. O algoritmo para calcular a área de interseção aproximada entre polígonos é apresentado na Figura 7.

```
1. REAL areaIntersecaoAproximada(assinat4CRS1, assinat4CRS2)
2.    areaAproximada = 0;
3.    MBRIntersecao = calculaMBRIntersecao(assinat4CRS1, assinat4CRS2);
4.    SE (assinat4CRS1.tamanhoAresta == assinat4CRS2.tamanhoAresta)
5.       menor4CRS = assinat4CRS1;
6.       maior4CRS = assinat4CRS2;
7.    SENAO
9.       menor4CRS = assMenorTamanho(assinat4CRS1, assinat4CRS2);
10.      maior4CRS = assMaiorTamanho(assinat4CRS1, assinat4CRS2);
11.   areaAproximada = 0;
12.    PARA CADA celula b de maior4CRS que está contida em MBRIntersecao
FACA
13.      PARA CADA celula s de menor4CRS que está contida em celula b FACA
14.         areaAproximada += areaEstimada[s.tipo,b.tipo];
15.   areaCelula = menor4CRS.tamanhoAresta * menor4CRS.tamanhoAresta;
16.   RETORNA areaAproximada * areaCelula;
```

Figura 7 - Algoritmo para calcular a área de interseção aproximada entre polígonos

Considere como exemplo, uma consulta que produza os seguintes pares de tipos de células 100 (*Pouco* × *Pouco*), 40 (*Pouco* × *Muito*), 70 (*Pouco* × *Cheio*), 60 (*Muito* × *Muito*) e 200 (*Cheio* × *Cheio*). Considere por simplificação que todas as células têm a mesma área, igual a 1. A área aproximada resultante desta consulta seria 265, como apresentado na Figura 8.

- $P \times P$: $100 \times 0{,}0625 = 6{,}25$
- $P \times M$: $40 \times 0{,}187 = 7{,}50$
- $P \times C$: $70 \times 0{,}2500 = 17{,}50$
- $M \times M$: $60 \times 0{,}5625 = 33{,}75$

- $C \times C$: $200 \times 1 = 200$ (*células cheio têm área exata!*)
- $Total$: 265

Figura 8 – Exemplo de cálculo da área de interseção aproximada

3.1.4 Algoritmo para calcular a União entre Assinaturas 4CRS

O algoritmo para calcular a assinatura correspondente à união de duas assinaturas 4CRS também corresponde a uma contribuição deste trabalho. O algoritmo calcula a assinatura de união entre duas assinaturas 4CRS da seguinte forma: se não existe MBR de interseção entre as assinaturas, então é retornado NULL. Por outro lado, quando existe MBR de interseção, uma nova assinatura 4CRS é criada, de acordo com o algoritmo apresentado na Figura 9. A simplificação de retornar NULL quando não há interseção entre assinaturas foi implementada por limitações no tamanho máximo do número de células da assinatura 4CRS e por esta tratar a assinatura de um polígono composto de apenas uma face, ou seja, a assinatura 4CRS implementada atualmente não contempla o armazenamento de polígonos com múltiplas faces como seria o caso, por exemplo, de armazenar os lagos existentes em um determinado local em uma única assinatura 4CRS.

```
1. ASSINAT4CRS uniaoAssinat4CRS(assinat4CRS1, assinat4CRS2)
2.  SE existeIntersecao(assinat4CRS1, assinat4CRS2)
3.    SE (assinat4CRS1.tamanhoAresta > assinat4CRS2.tamanhoAresta)
4.      maior4CRS = assinatat4CRS1;
5.                  menor4CRS   =   mudarResolucao(assinat4CRS2,   assi-
nat4CRS1.tamanhoAresta);
6.    SENAO
7.      maior4CRS = assinat4CRS2;
8.                  menor4CRS   =   mudarResolucao(assinat4CRS1,   assi-
nat4CRS2.tamanhoAresta);
9.    MBRUniao = calculaMBRUniao(menor4CRS.MBR, maior4CRS.MBR)
10.   /* Cria assinatura 4CRS com células Vazio */
11.   n4CRS = criaAssinatura(MBRUniao, maior4CRS.tamanhoAresta, VAZIO);
12.   PARA CADA b em maior4CRS.cels com intersecao com n em n4CRS.cels FACA
13.     n.tipo = b.tipo;
14.      PARA CADA s em menor4CRS.cels com intersecao com n em n4CRS.cels
FACA
15.       SE n.tipo == VAZIO OU s.tipo == CHEIO
16.         n.tipo = s.tipo;
17.       SENAO SE n.tipo == POUCO E s.tipo == MUITO
18.         n.tipo = s.tipo;
20.   RETORNA n4CRS;
21. SENAO
22.   RETORNA NULL;
```

Figura 9 – Algoritmo para calcular a união entre duas assinaturas 4CRS

3.2 Intervalos de confiança

Ao executar uma consulta que retorna resultados aproximados, é importante mostrar para o usuário um intervalo de confiança para a resposta obtida a fim de que ele possa decidir se a precisão é suficiente. A Equação 4 apresenta a fórmula para calcular o intervalo de confiança, proposta por Azevedo [2005], para os algoritmos a área aproximada e a área de interseção aproximada. Observe que, para realizar o cálculo, é necessário saber a média e a variância. A Seção 3.2.1 e a Seção

3.2.2 apresentam como calcular estes valores para cada um dos algoritmos, bem como apresentam um exemplo de cálculo do intervalo de confiança para cada caso.

$$Intervalo\ de\ confiança = \sum_n n_c \times \left(\mu_c \pm p \times \sqrt{\frac{\sigma_c^2}{n_c}} \right)$$

Onde:

c: tipo de célula ou combinação de tipos de células dependendo do algoritmo sendo avaliado;

μ_c: média (ou área esperada);

σ_c^2: variância;

p: intervalo de confiança escolhido, i.e., 1,96 para um intervalo de confiança de 95%;

n_c: número de células.

Equação 4 – Equação para calcular o intervalo de confiança

A partir dos cálculos de intervalo de confiança propostos por Azevedo [2005], neste trabalho, propomos uma forma de cálculo do intervalo de confiança para o algoritmo de similaridade, o qual é apresentado na Seção 3.2.3.

3.2.1 Cálculo do intervalo de confiança para o algoritmo que calcula a área aproximada

Como apresentado por Azevedo [2005], no caso do algoritmo que calcula a área aproximada, μ_c corresponde à área esperada para cada tipo de célula (Tabela 2, na Seção 3.1.2), n_c é o número de células de tipo c, e σ_c^2 é a variância correspondente ao tipo de célula c.

As variâncias para os tipos de células Vazio e Cheio são iguais a 0 (zero), já que estes tipos de células têm percentual exato de área do polígono dentro da célula, cujos valores são 0% e 100%, respectivamente. Por outro lado, é necessário estimar a variância para células dos tipos Pouco e Muito. Como dito anteriormente, assumindo que a distribuição de área do polígono dentro da célula é muito próxima da distribuição uniforme, a variância para células *Pouco* é de $(0{,}5\text{-}0)^2/12 = 1/48 = 0.020833$, já que células pouco têm distribuição no intervalo (0; 0,50]. Células *Muito* têm distribuição com a mesma variância $(1\text{-}0{,}5)^2/12 = 1/48 = 0.020833$ no intervalo (0,5; 1).

Azevedo [2005] apresenta um exemplo do cálculo do intervalo de confiança, o qual é descrito a seguir.

Um objeto possui a seguinte distribuição de células: 100 células *Pouco*, 120 células *Muito* e 400 células *Cheio*. Uma resposta com um intervalo de confiança de 95% pode ser calculada como é apresentado na Figura 10 (por simplicidade, assume-se que cada célula tem mesma área, com valor igual a 1).

- Células Pouco: $100 \times \left(0{,}25 \pm 1{,}96 \times \left(\frac{0{,}020833}{100}\right)^{\frac{1}{2}}\right) = 25 \pm 2{,}83$
- Células Muito: $120 \times \left(0{,}75 \pm 1{,}96 \times \left(\frac{0{,}020833}{120}\right)^{\frac{1}{2}}\right) = 90 \pm 3{,}10$
- Células Cheio: 400 (*células cheio possuem área exata!*)
- Total: $515 \pm 5{,}93$

Figura 10 – Exemplo de cálculo do intervalo de confiança de 95% para o algoritmo que calcula a área aproximada

Dessa forma, o intervalo de confiança tem uma variância de ±5,93. Isto significa que, para um intervalo de confiança de 95%, a área aproximada envolvendo estes números de células tem um erro de no máximo ±1,15%.

3.2.2 Cálculo do intervalo de confiança para o algoritmo que calcula a área de interseção aproximada

Como apresentado em Azevedo [2005], o cálculo do intervalo de confiança para o algoritmo que calcula a área de interseção aproximada depende que sejam calculados os valores de média (μ_c) e variância (σ_c^2) das áreas esperadas correspondentes à sobreposição de duas células de mesmo tamanho. Os valores de média são apresentados na Tabela 3 (Seção 3.1.3), e os valores da variância são apresentados na Tabela 4. Os cálculos para chegar a estes valores são apresentados detalhadamente por Azevedo [2005].

Tabela 4 – Variância da área esperada correspondente à sobreposição de tipos de células

Tipos de células	Vazio	Pouco	Muito	Cheio
Vazio	0	0	0	0
Pouco	0	0,003038194	0,013454861	0,020833333
Muito	0	0,013454861	0,023871528	0,020833333
Cheio	0	0,020833333	0,020833333	0

Azevedo [2005] apresenta um exemplo do cálculo do intervalo de confiança, o qual é descrito a seguir.

Considere uma consulta que produza os seguintes pares de tipos de células 100 (*Pouco* × *Pouco*), 40 (*Pouco* × *Muito*), 70 (*Pouco* × *Cheio*), 60 (*Muito* × *Muito*) e 200 (*Cheio* × *Cheio*). Uma resposta com um intervalo de confiança de 95% é apresentada na Figura 11 (por simplicidade, assume-se que todas as células têm a mesma área, igual a 1).

- $P \times P: 100 \times \left(0{,}0625 \pm 1{,}96 \times \left(\frac{0{,}0030382}{100}\right)^{\frac{1}{2}}\right) = 6{,}25 \pm 1{,}0803$
- $P \times M: 40 \times \left(0{,}1875 \pm 1{,}96 \times \left(\frac{0{,}013454}{40}\right)^{\frac{1}{2}}\right) = 7{,}50 \pm 1{,}4378$
- $P \times C: 70 \times \left(0{,}2500 \pm 1{,}96 \times \left(\frac{0{,}020833}{70}\right)^{\frac{1}{2}}\right) = 17{,}50 \pm 2{,}3669$
- $M \times M: 60 \times \left(0{,}5625 \pm 1{,}96 \times \left(\frac{0{,}023872}{60}\right)^{\frac{1}{2}}\right) = 33{,}75 \pm 2{,}3457$
- $C \times C: 200 \times 1 = 200$ (*células cheio têm área exata!*)
- $Total: 265 \pm 7{,}2308$

Figura 11 – Exemplo de cálculo do intervalo de confiança de 95% para o algoritmo que calcula a área de interseção aproximada

Assim, a resposta tem uma variância de ±7,2308, o que significa que, para um intervalo de confiança de 95%, a resposta aproximada para uma consulta envolvendo números de células como apresentados no exemplo terá um erro máximo de ±2,7286%.

3.2.3 Cálculo do intervalo de confiança para o algoritmo que calcula a similaridade entre polígonos

Com base nos cálculos do intervalo de confiança do algoritmo que calcula a área aproximada e do algoritmo que calcula a área de interseção aproximada, propomos uma fórmula para calcular o intervalo de confiança para o algoritmo de similaridade (Equação 5).

$$IC_{Similaridade} = [limite_{inferior_{\cong}}, limite_{superior_{\cong}}]$$

Onde:

$$limite_{inferior_{\cong}} = \frac{limite_{inferior_{\cap}}}{limite_{superior_{\cup}}}$$

$$limite_{superior_{\cong}} = \frac{limite_{superior_{\cap}}}{limite_{inferior_{\cup}}}$$

$$limite_{inferior_{\cap}} = A_{\cap} - \Delta_{IC_{\cap}}$$

$$limite_{superior_{\cap}} = A_{\cap} + \Delta_{IC_{\cap}}$$

$$limite_{inferior_{\cup}} = A_{\cup} - \Delta_{IC_{\cup}}$$

$$limite_{superior_{\cup}} = A_{\cup} + \Delta_{IC_{\cup}}$$

Equação 5 – Definição dos limites para a equação para o cálculo do intervalo de confiança

Logo, nossa proposta é a seguinte fórmula para o intervalo de confiança da similaridade:

$$IC_{Similaridade} = \left[\frac{A_{\cap} - \Delta_{IC_{\cap}}}{A_{\cup} + \Delta_{IC_{\cup}}}, \frac{A_{\cap} + \Delta_{IC_{\cap}}}{A_{\cup} - \Delta_{IC_{\cup}}}\right]$$

Onde:

- $A_{\cap}$: *área de interseção aproximada entre polígonos*
- $A_{\cup}$: *área aproximada da união entre polígonos*
- $\Delta_{IC_{\cap}}$: *variância do intervalo de confiança do algoritmo que calcula a área de interseção aproximada*
- $\Delta_{IC_{\cup}}$: *variância do intervalo de confiança do algoritmo que calcula a área aproximada da união*

Equação 6 – Equação para calcular o intervalo de confiança para o algoritmo que calcula a similaridade entre polígonos

Um exemplo de cálculo é demonstrado a seguir.

Considere que o algoritmo seja executado em dois polígonos quaisquer e obtenha os seguintes dados:

- $A_{\cap}$: $4{,}95 \times 10^6$
- $A_{\cup}$: $1{,}27 \times 10^7$
- $\Delta_{IC_{\cap}}$: $2{,}37 \times 10^5$
- $\Delta_{IC_{\cup}}$: $2{,}45 \times 10^5$

Aplicando a fórmula para cálculo do intervalo de confiança na resposta do algoritmo que calcula a similaridade entre polígonos, vamos obter o seguinte resultado:

$$IC_{Similaridade} = \left[\frac{4{,}95 \times 10^6 - 2{,}37 \times 10^5}{1{,}27 \times 10^7 + 2{,}45 \times 10^5}, \frac{4{,}95 \times 10^6 + 2{,}37 \times 10^5}{1{,}27 \times 10^7 - 2{,}45 \times 10^5}\right]$$

$$IC_{Similaridade} = \left[\frac{4{,}72 \times 10^6}{1{,}30 \times 10^7}, \frac{5{,}19 \times 10^6}{1{,}25 \times 10^7}\right]$$

$$IC_{Similaridade} = [0{,}364\,,\, 0{,}416]$$

4 Implementação da proposta no Secondo

Os algoritmos apresentados neste trabalho foram implementados no SECONDO, um banco de dados extensível utilizado para aprendizado de técnicas de banco de dados e realização de experimentos. A seção 4.1 apresenta as principais características do SECONDO, enquanto a seção 4.2 apresenta os algoritmos implementados.

4.1 SECONDO

SECONDO [Dieker and Güting, 2000; Güting *et al.*, 2005] é um ambiente genérico que suporta a implementação de sistemas de banco de dados para grande número de modelos de dados e linguagens de consulta. Ele é desenvolvido como um protótipo de pesquisa na Fernuniversität em Hagen. Tipos de dados, construtores desses tipos de dados e operadores são implementados em álgebras. Cada álgebra é baseada no conceito de assinatura de segunda ordem [Güting, 1993]: a primeira assinatura descreve os construtores dos tipos de dados e a segunda assinatura descreve operações sobre estes tipos. Uma álgebra pode ser instalada no SECONDO sem modificar a parte central do seu código. Depois de recompilar o SECONDO, a álgebra recém-adicionada pode ser utilizada.

O objetivo do SECONDO é prover um *framework* "genérico" de um sistema de banco de dados que possa ser preenchido com a implementação de diversos modelos de dados de SGBDs. Por exemplo, é possível implementar modelos relacionais, orientado-a-objeto, temporais ou XML para acomodar tipos de dados para dados espaciais, de objetos em movimento, de fórmulas químicas, etc.

Originalmente, o SECONDO destinava-se a ser uma plataforma para implementação e experimentação de novos tipos de modelos de dados, especialmente para apoiar modelos de banco de dados espaciais, espaço-temporais e de grafos. No entanto, devido ao SECONDO ter uma arquitetura de fácil entendimento, e ao mesmo tempo simples e sofisticada, Güting *et al.* [2005] enfatizam que, desde que todo o código-fonte esteja acessível e compreensível em grande parte pelos estudantes, ele é também uma excelente ferramenta para ensinar sobre arquitetura de banco de dados e conceitos de implementação.

O SECONDO compreende três principais componentes escritos em diferentes linguagens:

- O kernel: implementa modelos de dados específicos, é extensível pelos módulos de álgebras e provê processamento de consultas sobre as álgebras implementadas. Ele é escrito em C++.
- O otimizador: fornece a otimização de consultas conjuntivas para um ambiente relacional e implementa a parte essencial das linguagens *SQL-like*, em uma notação adaptada para PROLOG. Ele é escrito em PROLOG.

- A interface gráfica com o usuário (Graphical User Interface – GUI): provê uma interface extensível para novos tipos ou modelos de dados. Ela é escrita em Java.

Os três componentes podem ser utilizados juntos ou não, de diversas formas. A Figura 12 mostra uma configuração onde eles são utilizados juntos. Nesta configuração, a GUI pode pedir ao *kernel* diretamente para executar os comandos e consultas ou pode chamar o otimizador para obter um plano para uma dada consulta SQL. O otimizador quando necessário chama o *kernel* para obter informações sobre os esquemas e cardinalidades das relações e seletividade dos predicados.

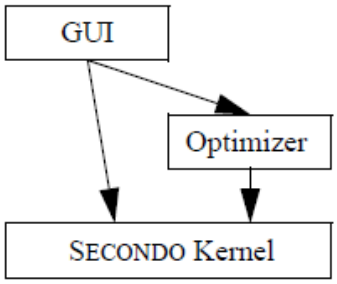

Figura 12 – Colaboração entre os componentes do SECONDO

Nos testes experimentais realizados neste trabalho, o otimizador não foi utilizado. A interface mono-usuário SecondoTTYBDB do SECONDO foi utilizada para realização das consultas e operações no SECONDO. A GUI foi utilizada para visualizar objetos e para análise de resultados obtidos.

4.2 Implementação dos algoritmos

No SECONDO, foram implementados os operadores de similaridade, área aproximada e área de interseção aproximada, união de assinaturas 4CRS e interseção de assinaturas 4CRS. Eles foram implementados na álgebra Raster, a qual implementa os tipos de dados correspondentes à assinatura 4CRS, seus construtores de tipo e operadores.

A instalação disponível no site do SECONDO[1] não inclui o registro da Álgebra Raster no sistema. Dessa forma, é necessário vinculá-la. Os passos necessários para realizar este registro são apresentados no Apêndice 1.

Para implementar operadores em uma álgebra no SECONDO é necessário seguir os passos detalhados no Apêndice 2. Os códigos-fonte dos algoritmos implementados são apresentados no Apêndice 3.

A Figura 13 apresenta a especificação do método similaridade que é apresentada para o usuário que acessa o banco de dados. Esta especificação contém:

- *Nome do método:* O método similaridade foi criado no SECONDO com o nome *rSimilar*.
- *Assinatura:* o método recebe como entrada duas assinaturas 4CRS e retorna um objeto *approxresult*. O objeto *approxresult* contém as seguintes

[1] http://dna.fernuni-hagen.de/Secondo.html/

informações: resultado aproximado e resultado mínimo e resultado máximo de acordo com o intervalo de confiança de definido%.

- *Sintaxe:* para utilizar o método é necessário digitar o primeiro argumento (que deve ser uma assinatura 4CRS), depois *rSimilar*, e, em seguida, o segundo argumento (que também deve ser uma assinatura 4CRS).
- *Significado:* explicação do que o método faz.
- *Exemplo:* exemplo de uso do método.

```
       Nome: rSimilar
 Assinatura: (Raster4CRS, Raster4CRS) -> approxresult
    Sintaxe: _ rSimilar _
Significado: Retorna o percentual de similaridade entre duas assinaturas
             4CRS junto com seu limite de erro.
    Exemplo: query raster4CRS1 rSimilar raster4CRS2
```

Figura 13 - Especificação do operador rSimilar no SECONDO

A Figura 14 apresenta a especificação dos demais métodos e dos construtores de tipo da álgebra Raster. Estas informações são obtidas através do comando *list algebra RasterAlgebra*, no SECONDO.

```
------------------------------
Algebra : RasterAlgebra
------------------------------
--------------------
  Construtores de tipo
--------------------
             Nome: raster4CRS0
       Assinatura: -> DATA
 Tipos de exemplo: raster4CRS
    Rep. da Lista: (<id> <type> <min.x> <min.y> <max.x> <max.y>
                    <sizeOfBlock> <dx> <dy> (<weight[i]>*))
      Comentários: todos os valores devem ser do tipo int.

             Nome: approxresult
       Assinatura: -> DATA
 Tipos de exemplo: approxresult
     Rep da Lista: (<result> <resultMin> <resultMax>)
 Exemplo de lista: (100.0 90.2 109.8)
      Comentários: todos os valores devem ser do tipo real.

--------------------
  Operadores
--------------------
             Nome: calc4CRS
       Assinatura: region -> raster4CRS
          Sintaxe: calc4CRS (_)
      Significado: Calcula a 4CRS.
          Exemplo: query calc4CRS (testRegion)

             Nome: calc3CRS
       Assinatura: (region || line || points) -> Raster4CRS
          Sintaxe: Calc3CRS (_)
      Significado: Calcula a 3CRS.
          Exemplo: query calc3CRS (testRegion)

             Nome: rIntersects
       Assinatura: (Raster4CRS Raster4CRS) -> int
          Sintaxe: _ rIntersects _
      Significado: Indica quando duas assinaturas 4CRS têm interseção.
          Exemplo: query raster4CRS1 rIntersects raster4CRS2

             Nome: rSimilar
```

```
 Assinatura: (Raster4CRS, Raster4CRS) -> real
    Sintaxe: _ rSimilar _
Significado: Retorna o percentual de similaridade entre duas
             assinaturas 4CRS junto com seu limite de erro.
    Exemplo: query raster4CRS1 rSimilar raster4CRS2

       Nome: rArea
 Assinatura: (Raster4CRS) -> float
    Sintaxe: rArea (_)
Significado: Retorna a área aproximada de uma 4CRS.
    Exemplo: query rArea (raster4CRS1)

       Nome: rPlus
 Assinatura: (Raster4CRS, Raster4CRS) -> Raster4CRS
    Sintaxe: _ rPlus _
Significado: Retorna a assinatura 4CRS correspondente à união entre
             duas assinaturas 4CRS.
    Exemplo: query raster4CRS1 rPlus raster4CRS2

       Nome: rIntersection
 Assinatura: (Raster4CRS, Raster4CRS) -> Raster4CRS
    Sintaxe: _ rIntersection _
Significado: Retorna a 4CRS correspondente à interseção entre duas
             assinaturas 4CRS.
    Exemplo: query raster4CRS1 rIntersection raster4CRS2

       Nome: rOverlappingArea
 Assinatura: (Raster4CRS, Raster4CRS) -> real
    Sintaxe: rOverlappingArea (_,_)
Significado: Retorna a área de interseção aproximada entre duas
             assinaturas 4CRS.
    Exemplo: query rOverlappingArea(raster4CRS1, raster4CRS2)
```

Figura 14 – Especificação dos construtores de tipo e operadores da álgebra Raster

5 Testes experimentais

Foram realizados testes experimentais com o objetivo de avaliar a precisão da resposta do algoritmo para cálculo aproximado da similaridade em relação à resposta exata. Os testes foram executados em um PC Core 2 Duo 2.66 GHz com 2 GB de memória RAM.

5.1 Preparação do ambiente

Nos testes experimentais, foram utilizados conjuntos de dados compostos por polígonos que representam municípios da região norte do Brasil (*BRNorte*). Este conjunto foi obtido a partir de um conjunto de dados com todos os municípios do Brasil (Figura 15), realizando a consulta apresentada na Figura 16 que segue a notação do Secondo. Neste consulta, são computados somente os municípios que têm interseção com o retângulo *Rect1*, destacado na Figura 17, os quais são armazenados no objeto *BRNorte*. O conjunto *BRNorte* é apresentado na Figura 18.

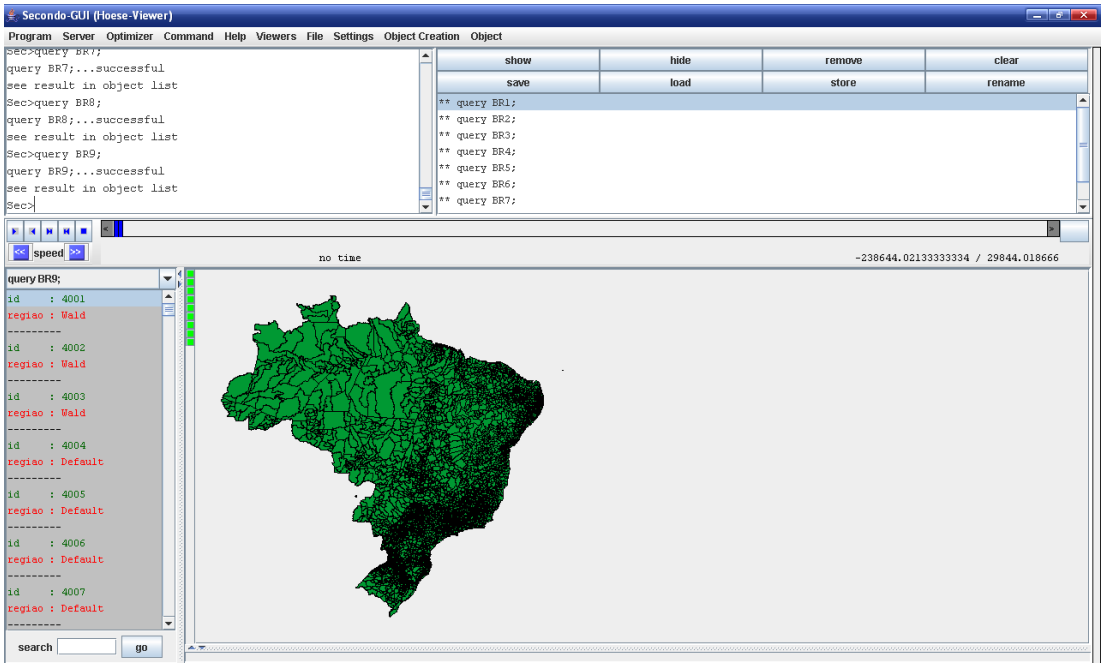

Figura 15 – Conjunto de dados com todos os municípios do Brasil

```
let BRNorte = BR feed filter [.regiao intersects rect1 rect2region] consume;
```

Figura 16 – Consulta para construir *BRNorte*

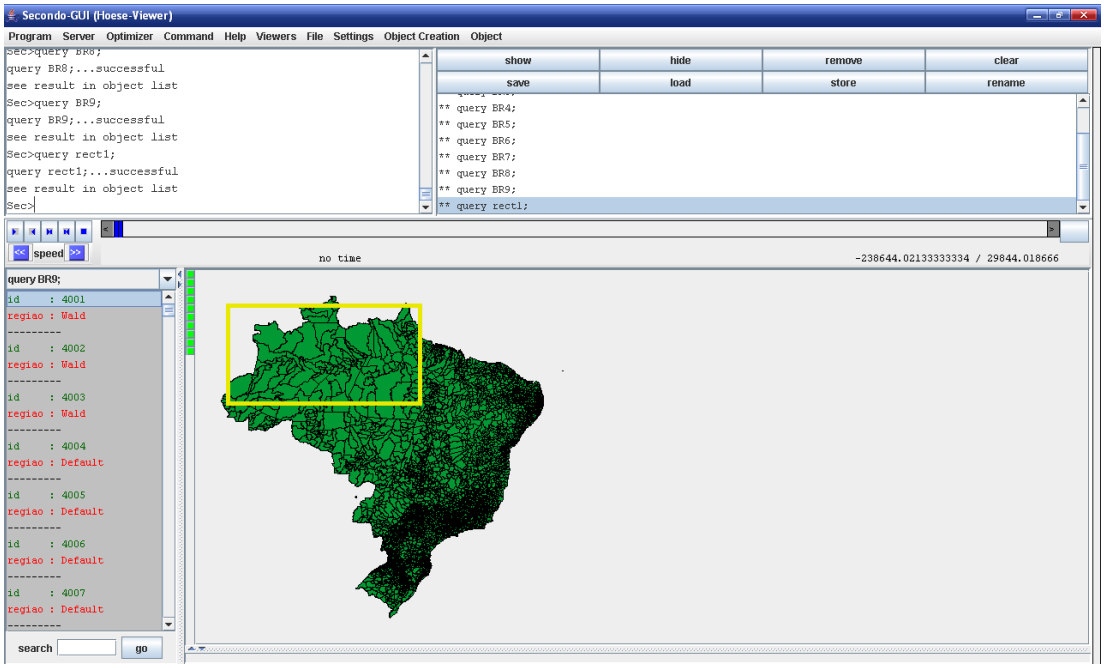

Figura 17 – Retângulo usado para construir *BRNorte*, destacado no mapa do Brasil

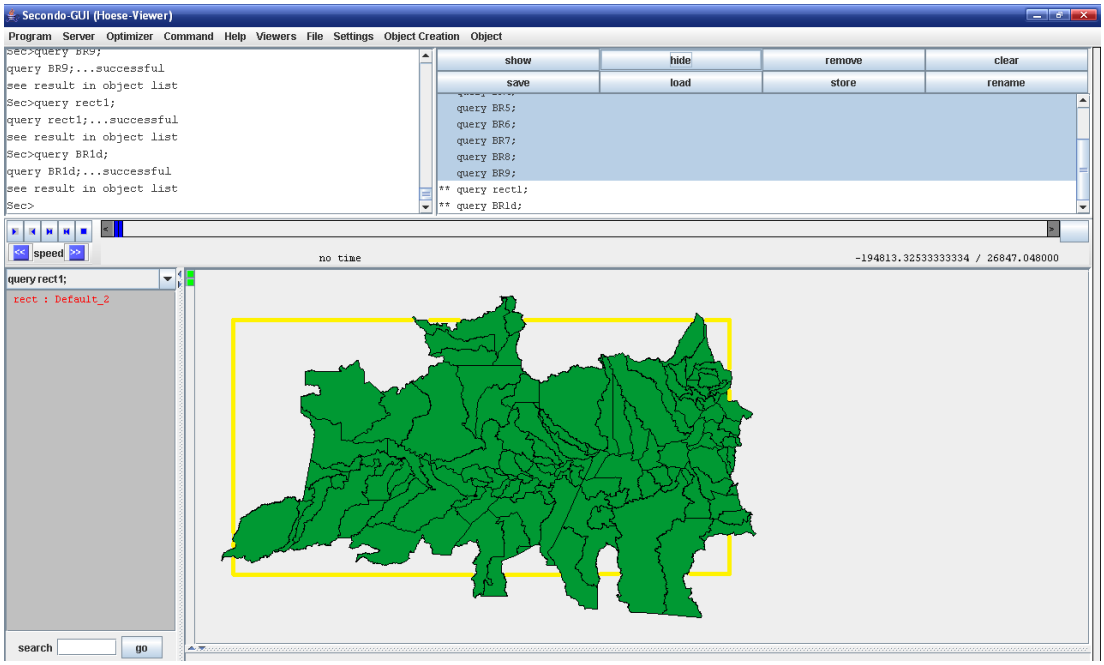

Figura 18 – Conjunto de dados *BRNorte*

Com o objetivo de obter outro conjunto de dados com sobreposição com *BRNorte*, para avaliarmos o operador de similaridade, os polígonos originais de *BRNorte* foram deslocados aleatoriamente nas coordenadas *x* e *y*, seguindo a proposta de Brinkhoff *et al.* [1994]. Foi construído um novo conjunto de dados chamado de *BRNorteT*, a partir da execução consulta apresentada na Figura 19. Este novo conjunto de dados é apresentado na Figura 20. Os dois conjuntos sobrepostos são apresentados na Figura 21.

```
let BRNorteT =
    BR1Norte feed extend [idT: .id, regiaoT: .regiao translate [ifthenelse(
    randint(10)<5, int2real(randint(8380)), int2real( -1 * randint(8380)) ),
    ifthenelse(randint(10)<5, int2real(randint(4265)), int2real(-1 * randint(4265))
    )]] remove[id] remove[regiao]
consume;
```

Figura 19 – Consulta para construir *BRNorteT*

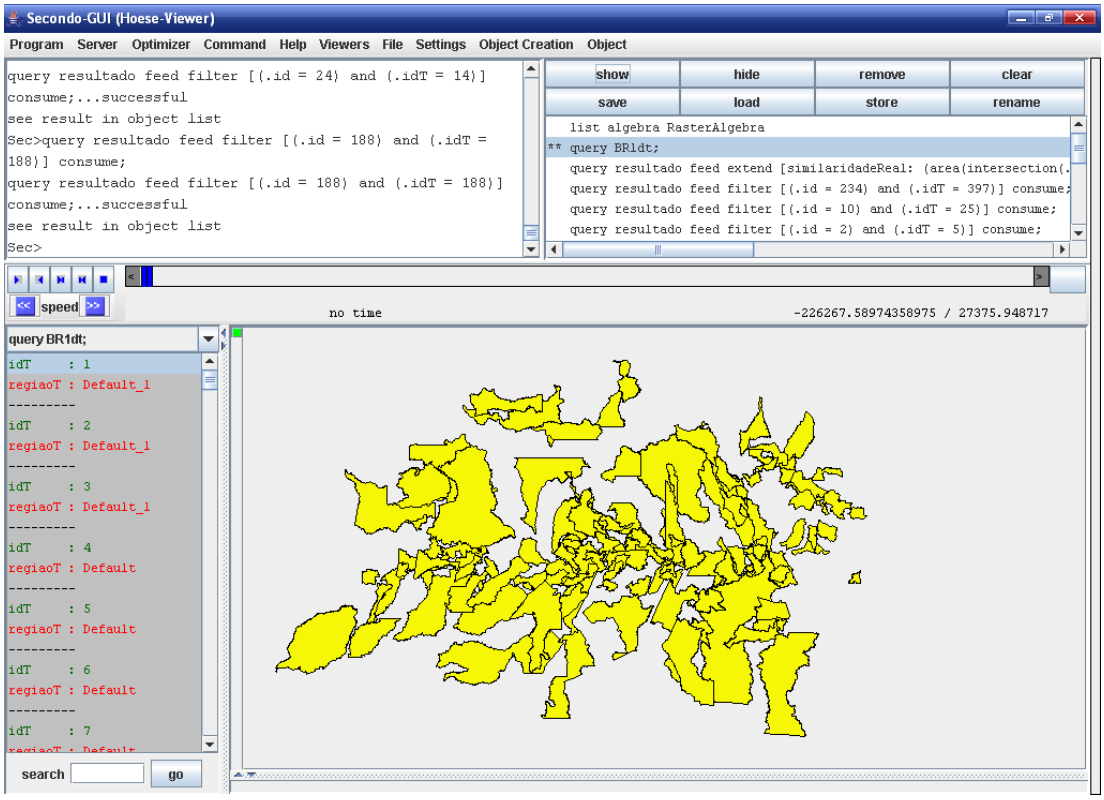

Figura 20 – Conjunto de dados *BRNorteT*

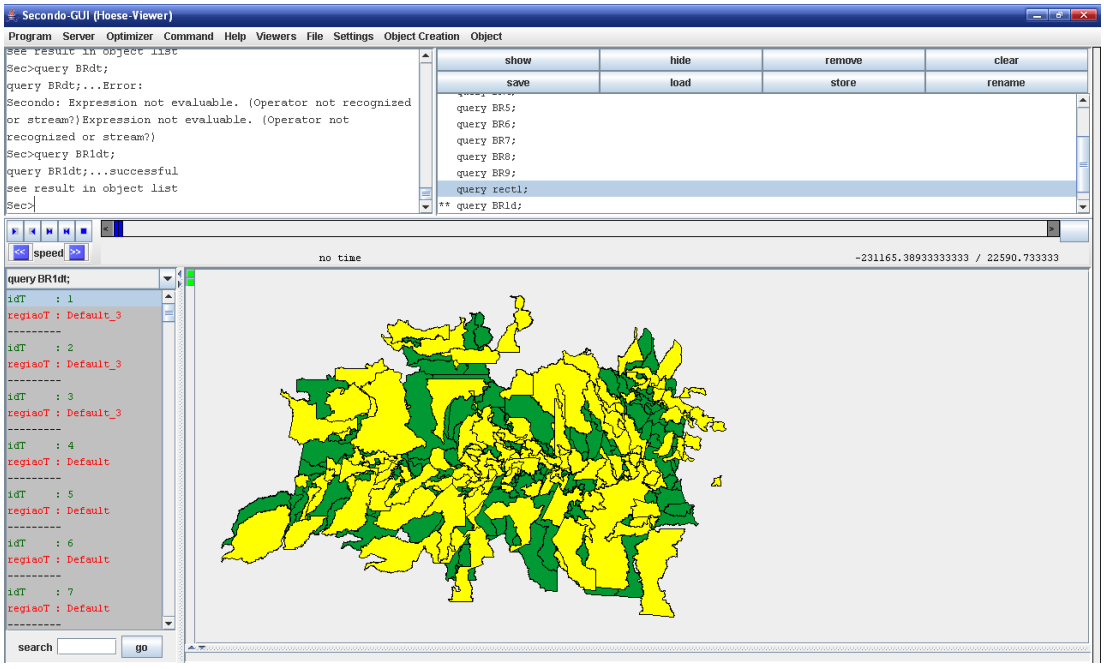

Figura 21 – Conjuntos de dados *BRNorte* e *BRNorteT* sobrepostos

A partir destes dois conjuntos foram geradas as assinaturas 4CRS para cada um dos seus objetos com a invocação do método *calc4CRS* da álgebra Raster, apresentado na Figura 22. O objeto *rasterBRNorte* foi estendido para conter um novo atributo que corresponde à assinatura Raster do atributo *regiao*. O objeto *rasterBRNorteT*

foi estendido para conter um novo atributo correspondente à assinatura Raster do atributo *regiaoT*.

Vale ressaltar que a visualização das assinaturas 4CRS correspondentes aos polígonos não são apresentadas, pois esta funcionalidade não foi implementada na interface gráfica com o usuário (Javagui) do Secondo.

```
let rasterBRNorte = BRNorte feed extend[rRegiao: calc4CRS(.regiao)] consume;
let rasterBRNorteT = BRNorteT feed extend[rRegiaoT: calc4CRS(.regiaoT)] consume;
```

Figura 22 – Consultas para computar as assinaturas 4CRS de BRNorte e BRNorteT

5.2 Execução dos testes

Como o objetivo dos testes era avaliar a precisão do algoritmo de similaridade, foi considerado, por simplificação, a execução do algoritmo apenas para os objetos de *BRNorte* e *BRNorteT* que possuíam interseção, como apresentado na Figura 23, através da consulta apresentada na Figura 24. Este comando atribui ao objeto *resultado* os registros de *rasterBRNorte* e *rasterBRNorteT*que têm interseção dos atributos *região* e *regiaoT*.

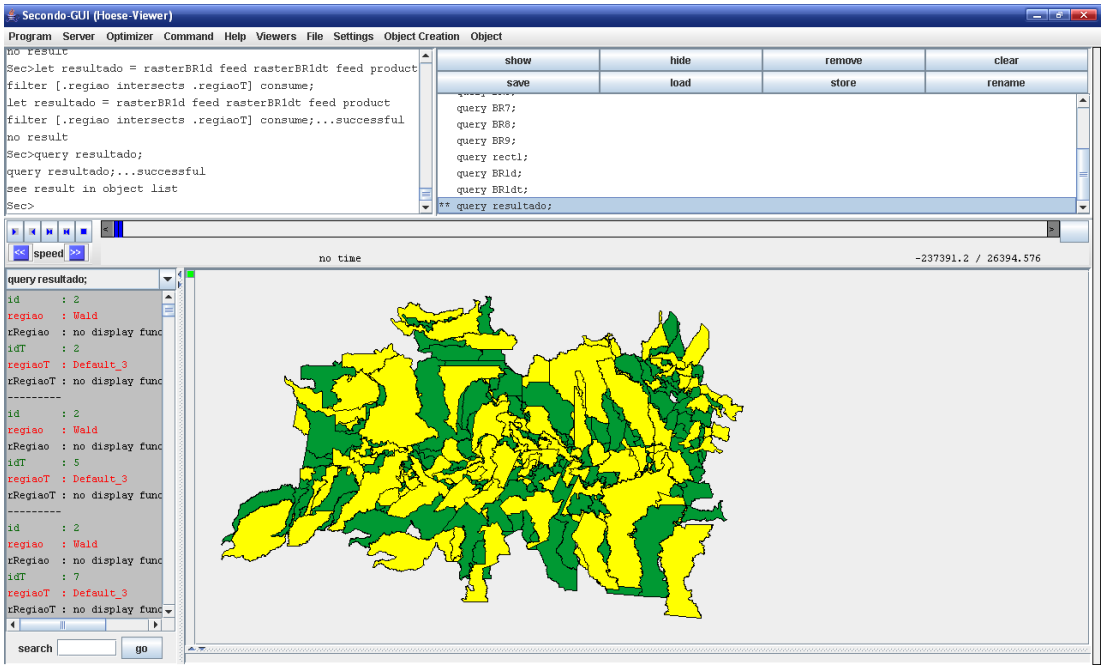

Figura 23 - Objetos de *BRNorte* e *BRNorteT* que têm interseção entre si

```
let resultado = rasterBRNorte feed rasterBRNorteT feed product filter [.regiao intersects .regiaoT] consume;
```

Figura 24 – Consulta para computar os pares de objetos com interseção entre si

Foram então calculadas a similaridade a partir dos objetos reais (utilizando os valores exatos da área de interseção e área de união), e a similaridade a partir de suas respectivas assinaturas 4CRS (utilizando os valores aproximados da área de interseção e da área de união). Os resultados obtidos foram apresentados considerando dois casos "resultados com erros abaixo de 10%" e "resultados com erros

acima de 10%". Devido a limitações de espaço para colocar todos os resultados em uma única tabela, os resultados correspondentes a cada caso são apresentados em duas tabelas. A Tabela 5 e a Tabela 6 apresentam os resultados com erros acima de 10%, enquanto que a Tabela 7 e a Tabela 8 apresentam os resultados com erros abaixo de 10%.

Os rótulos das colunas da Tabela 5, Tabela 6, Tabela 7 e Tabela 8 são definidos a seguir:

- ***ID1***: Identificação do objeto pertencente à *BRNorte*.
- ***ID2***: Identificação do objeto pertencente à *BRNorteT*.
- ***SB1***: Tamanho da célula do objeto de *BRNorte*.
- ***SB2***: Tamanho da célula do objeto de *BRNorteT*.
- ***AIR***: Área de interseção real entre os objetos.
- ***AIA***: Área de interseção aproximada entre os objetos.
- ***%EAIA***: Porcentagem de erro da área de interseção aproximada.
- ***MIN***: Valor inferior fornecido pelo intervalo de confiança da área de interseção aproximada.
- ***MAX***: Valor superior fornecido pelo intervalo de confiança da área de interseção aproximada.
- ***NC***: Número de células com interseção entre as assinaturas, exceto interseções com células *Vazio*. Ou seja, são descartadas interseções Vazio × Vazio, Vazio × Pouco, Vazio × Muito e Vazio × Cheio, pois estes casos não influenciam na nossa análise.
- ***AUR***: Área de união real entre os objetos.
- ***AUA***: Área de união aproximada entre os objetos.
- ***MIN2***: Valor inferior fornecido pelo intervalo de confiança da área de união aproximada.
- ***MAX2***: Valor superior fornecido pelo intervalo de confiança da área de união aproximada.
- ***%EAUA***: Porcentagem de erro da área de união aproximada.
- ***SRaster***: Porcentagem de similaridade entre os objetos calculada através de assinaturas raster.
- ***MIN3***: Valor inferior fornecido pelo intervalo de confiança da similaridade raster.
- ***MAX3***: Valor superior fornecido pelo intervalo de confiança da similaridade raster.
- ***SReal***: Porcentagem de similaridade entre os objetos calculada através dos objetos reais.
- ***%ES***: Porcentagem de erro da similaridade raster. Calculada de acordo com a Equação 7.

- ***%DE***: Porcentagem referente à diferença absoluta entre a similaridade real e a similaridade raster.

$$\%ES = \frac{|\ SRaster - SReal\ |}{SReal}$$

Equação 7 – Equação para calcular o erro de cálculo da similaridade

Tabela 5 – Tabela de resultados com erros acima de 10% (primeira parte)

	ID1	ID2	SB1	SB2	AIR	AIA	%EAIA	MIN	MAX	NC
1	**234**	**397**	**256**	**1024**	**124,011**	**312895**	**252212,29%**	**283097**	**342693**	**2**
2	**10**	**25**	**1024**	**256**	**486,325**	**1,05E+06**	**214822,12%**	**137623**	**1,95E+06**	**4**
3	274	164	512	256	1315,56	21286,1	1518,03%	-18765,4	61337,6	2
4	28	32	512	512	28746,6	248775	765,41%	6657,73	490892	10
5	167	188	256	1024	147499	1,00E+06	579,77%	961731	1,04E+06	3
6	146	146	1024	1024	301802	1,02E+06	236,81%	233996	1,80E+06	6
7	4	25	512	256	44569,8	127979	187,14%	-44221	300178	4
8	188	167	1024	256	75920,8	199020	162,14%	-152650	550689	2
9	139	38	256	512	9513,49	21286,1	123,75%	11273,2	31299	2
10	**2**	**5**	**1024**	**512**	**5,14E+06**	**1,07E+07**	**108,93%**	**7,90E+06**	**1,36E+07**	**26**
11	**140**	**146**	**1024**	**1024**	**613326**	**1,26E+06**	**105,12%**	**325925**	**2,19E+06**	**9**
12	301	414	512	1024	441836	774898	75,38%	655461	894335	3
13	130	276	512	1024	146264	241592	65,18%	141944	341240	3
14	78	230	512	256	76525,4	127979	67,24%	-44221	300178	4
15	276	139	1024	256	1,31E+06	2,09E+06	59,59%	654776	3,52E+06	7
16	**183**	**44**	**256**	**1024**	**1,60E+06**	**2,46E+06**	**53,03%**	**2,36E+06**	**2,55E+06**	**10**
17	**3**	**15**	**512**	**256**	**558978**	**287598**	**48,55%**	**-30710,4**	**605907**	**10**
18	114	168	512	256	488766	255381	47,75%	-55761,1	566522	6
19	12	109	512	128	457453	266312	41,78%	-26689,7	559314	8
20	22	27	128	256	71947,9	102498	42,46%	76771	128226	8
21	275	44	256	1024	1,91E+06	2,67E+06	39,19%	2,58E+06	2,75E+06	8
22	153	153	512	512	234690	321126	36,83%	5502,96	636750	5
23	205	153	256	512	1,15E+06	1,51E+06	31,51%	1,35E+06	1,66E+06	18
24	8	31	1024	128	1,67E+06	2,24E+06	34,52%	1,59E+06	2,90E+06	5
25	351	331	512	1024	4,83E+06	3,48E+06	28,07%	3,00E+06	3,96E+06	16
26	40	35	256	256	188064	243964	29,72%	82988	404941	18
27	13	14	1024	1024	1,21E+06	1,57E+06	29,99%	126816	3,01E+06	10
28	377	134	1024	512	3,55E+06	4,78E+06	34,56%	3,25E+06	6,32E+06	11
29	152	209	512	256	1,99E+06	1,47E+06	26,10%	991185	1,95E+06	14
30	158	167	512	256	1,46E+06	1,15E+06	20,71%	733032	1,57E+06	14
31	114	116	512	256	2,03E+06	1,60E+06	21,13%	1,10E+06	2,10E+06	15
32	152	280	512	128	968948	754686	22,11%	413206	1,10E+06	9
33	78	173	512	256	4,11E+06	3,31E+06	19,38%	2,65E+06	3,98E+06	30
34	4	15	512	256	3,35E+06	2,67E+06	20,31%	2,18E+06	3,15E+06	23
35	24	36	1024	1024	334453	398039	19,01%	-238940	1,04E+06	4
36	155	78	256	512	511564	420164	17,87%	350493	489836	7
37	16	4	256	512	126499	148976	17,77%	99352,9	198600	5
38	146	142	1024	512	5,72E+06	4,95E+06	13,46%	3,12E+06	6,79E+06	14
39	153	135	512	256	1,69E+06	1,96E+06	15,91%	1,35E+06	2,57E+06	19
40	276	480	1024	512	1,16E+07	1,34E+07	16,19%	1,10E+07	1,58E+07	27
41	109	21	128	512	310907	349595	12,44%	332811	366379	5
42	78	38	512	512	7,45E+06	6,52E+06	12,46%	5,72E+06	7,32E+06	54
43	9	21	1024	512	9,37E+06	8,14E+06	13,13%	5,89E+06	1,04E+07	18
44	139	78	256	512	3,47E+06	3,10E+06	10,88%	2,97E+06	3,22E+06	24
45	188	276	1024	1024	1,24E+07	1,38E+07	11,87%	1,06E+07	1,70E+07	39
46	146	152	1024	512	7,61E+06	6,84E+06	10,08%	4,81E+06	8,88E+06	16
47	275	397	256	1024	3,66E+06	3,18E+06	13,16%	3,08E+06	3,28E+06	11
48	138	163	256	256	96603,1	86710,7	10,24%	-8720,8	182142	9
49	115	115	128	256	332498	366248	10,15%	335277	397219	15
50	209	97	256	512	240774	266312	10,61%	190882	341742	8

Tabela 6 – Tabela de resultados com erros acima de 10% (segunda parte)

	ID1	ID2	AUR	AUA	MIN2	MAX2	%EAUA	SRaster	MIN3	MAX3	SReal	%ES	%DE
1	**234**	**397**	**5,53E+07**	**5,43E+07**	**5,09E+07**	**5,76E+07**	**1,96%**	**0,58%**	**0,49%**	**0,67%**	**0,00%**	**257252,94%**	**0,58%**
2	**10**	**25**	**4,21E+07**	**4,17E+07**	**3,82E+07**	**4,52E+07**	**1,04%**	**2,51%**	**0,30%**	**5,11%**	**0,00%**	**217074,82%**	**2,51%**
3	274	164	2,82E+07	2,82E+07	2,72E+07	2,92E+07	0,01%	0,08%	-0,06%	0,23%	0,00%	1518,24%	0,07%
4	28	32	3,50E+07	3,53E+07	3,42E+07	3,63E+07	0,70%	0,71%	0,02%	1,43%	0,08%	759,35%	0,62%
5	167	188	1,79E+08	1,79E+08	1,75E+08	1,83E+08	0,09%	0,56%	0,53%	0,60%	0,08%	580,39%	0,48%
6	146	146	1,47E+08	1,46E+08	1,42E+08	1,51E+08	0,27%	0,69%	0,15%	1,27%	0,21%	237,73%	0,49%
7	4	25	2,19E+07	2,18E+07	2,07E+07	2,28E+07	0,55%	0,59%	-0,19%	1,45%	0,20%	188,73%	0,38%
8	188	167	1,79E+08	1,81E+08	1,77E+08	1,86E+08	1,33%	0,11%	-0,08%	0,31%	0,04%	158,70%	0,07%
9	139	38	1,87E+07	1,80E+07	1,69E+07	1,90E+07	3,89%	0,12%	0,06%	0,19%	0,05%	132,80%	0,07%
10	**2**	**5**	**6,87E+07**	**6,87E+07**	**6,51E+07**	**7,22E+07**	**0,04%**	**15,63%**	**10,93%**	**20,84%**	**7,48%**	**108,86%**	**8,15%**
11	**140**	**146**	**1,33E+08**	**1,32E+08**	**1,27E+08**	**1,36E+08**	**1,32%**	**0,96%**	**0,24%**	**1,72%**	**0,46%**	**107,88%**	**0,50%**
12	301	414	9,02E+07	9,10E+07	8,69E+07	9,50E+07	0,83%	0,85%	0,69%	1,03%	0,49%	73,94%	0,36%
13	130	276	1,01E+08	9,96E+07	9,53E+07	1,04E+08	1,77%	0,24%	0,14%	0,36%	0,14%	68,15%	0,10%
14	78	230	1,77E+07	1,80E+07	1,71E+07	1,89E+07	1,96%	0,71%	-0,23%	1,75%	0,43%	64,02%	0,28%
15	276	139	7,08E+07	7,16E+07	6,76E+07	7,55E+07	1,14%	2,92%	0,87%	5,21%	1,85%	57,79%	1,07%
16	**183**	**44**	**4,05E+07**	**4,12E+07**	**3,82E+07**	**4,41E+07**	**1,58%**	**5,97%**	**5,36%**	**6,67%**	**3,96%**	**50,65%**	**2,01%**
17	**3**	**15**	**3,30E+07**	**3,35E+07**	**3,25E+07**	**3,45E+07**	**1,60%**	**0,86%**	**-0,09%**	**1,87%**	**1,70%**	**49,36%**	**0,84%**
18	114	168	3,30E+07	3,32E+07	3,22E+07	3,42E+07	0,54%	0,77%	-0,16%	1,76%	1,48%	48,03%	0,71%
19	12	109	2,96E+07	3,08E+07	2,98E+07	3,19E+07	3,92%	0,86%	-0,08%	1,88%	1,54%	43,98%	0,68%
20	22	27	6,79E+06	6,93E+06	6,65E+06	7,21E+06	2,12%	1,48%	1,06%	1,93%	1,06%	39,50%	0,42%
21	275	44	4,45E+07	4,51E+07	4,19E+07	4,83E+07	1,38%	5,91%	5,35%	6,56%	4,31%	37,30%	1,61%
22	153	153	4,07E+07	4,08E+07	3,96E+07	4,19E+07	0,27%	0,79%	0,01%	1,61%	0,58%	36,47%	0,21%
23	205	153	2,55E+07	2,50E+07	2,41E+07	2,60E+07	1,98%	6,03%	5,22%	6,90%	4,49%	34,16%	1,54%
24	8	31	7,68E+07	7,84E+07	7,51E+07	8,16E+07	2,02%	2,86%	1,94%	3,86%	2,17%	31,86%	0,69%
25	351	331	8,52E+07	8,78E+07	8,45E+07	9,11E+07	3,12%	3,96%	3,29%	4,68%	5,68%	30,25%	1,72%
26	40	35	1,60E+07	1,59E+07	1,56E+07	1,62E+07	0,37%	1,53%	0,51%	2,59%	1,18%	30,20%	0,36%
27	13	14	2,43E+08	2,44E+08	2,39E+08	2,48E+08	0,29%	0,64%	0,05%	1,26%	0,50%	29,61%	0,15%
28	377	134	1,11E+08	1,16E+08	1,12E+08	1,20E+08	4,68%	4,12%	2,70%	5,65%	3,20%	28,55%	0,91%
29	152	209	1,60E+07	1,62E+07	1,54E+07	1,70E+07	1,04%	9,10%	5,84%	12,69%	12,44%	26,85%	3,34%
30	158	167	3,00E+07	3,11E+07	3,01E+07	3,21E+07	3,65%	3,71%	2,29%	5,24%	4,85%	23,50%	1,14%
31	114	116	2,94E+07	3,00E+07	2,91E+07	3,08E+07	2,00%	5,34%	3,56%	7,23%	6,91%	22,67%	1,57%
32	152	280	1,54E+07	1,53E+07	1,45E+07	1,61E+07	0,67%	4,92%	2,56%	7,54%	6,28%	21,59%	1,36%
33	78	173	1,49E+07	1,51E+07	1,44E+07	1,59E+07	1,35%	21,90%	16,70%	27,65%	27,53%	20,46%	5,63%
34	4	15	2,13E+07	2,12E+07	2,03E+07	2,22E+07	0,52%	12,57%	9,84%	15,55%	15,69%	19,89%	3,12%
35	24	36	1,25E+08	1,25E+08	1,21E+08	1,29E+08	0,14%	0,32%	-0,19%	0,86%	0,27%	18,85%	0,05%
36	155	78	1,39E+07	1,39E+07	1,31E+07	1,47E+07	0,14%	3,02%	2,38%	3,75%	3,69%	17,98%	0,66%
37	16	4	2,49E+07	2,54E+07	2,44E+07	2,65E+07	2,10%	0,59%	0,37%	0,82%	0,51%	15,34%	0,08%
38	146	142	7,86E+07	8,00E+07	7,64E+07	8,35E+07	1,76%	6,19%	3,73%	8,89%	7,28%	14,96%	1,09%
39	153	135	2,55E+07	2,59E+07	2,50E+07	2,68E+07	1,42%	7,57%	5,02%	10,31%	6,63%	14,29%	0,95%
40	276	480	1,11E+08	1,13E+08	1,08E+08	1,17E+08	1,71%	11,91%	9,43%	14,59%	10,43%	14,24%	1,48%
41	109	21	2,89E+07	2,88E+07	2,78E+07	2,97E+07	0,61%	1,22%	1,12%	1,32%	1,07%	13,14%	0,14%
42	78	38	1,63E+07	1,63E+07	1,55E+07	1,71E+07	0,34%	39,95%	33,39%	47,20%	45,80%	12,76%	5,84%
43	9	21	1,31E+08	1,30E+08	1,26E+08	1,34E+08	1,09%	6,27%	4,41%	8,24%	7,14%	12,17%	0,87%
44	139	78	1,41E+07	1,43E+07	1,35E+07	1,51E+07	1,04%	21,67%	19,66%	23,93%	24,57%	11,79%	2,90%
45	188	276	2,24E+08	2,24E+08	2,20E+08	2,29E+08	0,27%	6,16%	4,64%	7,74%	5,52%	11,57%	0,64%
46	146	152	7,89E+07	8,00E+07	7,63E+07	8,36E+07	1,37%	8,56%	5,75%	11,65%	9,65%	11,30%	1,09%
47	275	397	5,41E+07	5,30E+07	4,98E+07	5,61E+07	2,12%	6,00%	5,48%	6,58%	6,76%	11,28%	0,76%
48	138	163	1,04E+07	1,04E+07	1,02E+07	1,07E+07	0,39%	0,83%	-0,08%	1,79%	0,93%	10,59%	0,10%
49	115	115	7,92E+06	7,90E+06	7,64E+06	8,15E+06	0,25%	4,64%	4,11%	5,20%	4,20%	10,42%	0,44%
50	209	97	2,26E+07	2,27E+07	2,17E+07	2,37E+07	0,34%	1,17%	0,81%	1,58%	1,07%	10,23%	0,11%

Tabela 7 – Tabela de resultados com erros abaixo de 10% (primeira parte)

	ID1	ID2	SB1	SB2	AIR	AIA	%EAIA	MIN	MAX
51	68	158	512	512	2,82E+06	3,09E+06	9,64%	2,43E+06	3,76E+06
52	**188**	**379**	**1024**	**256**	**1,15E+07**	**1,28E+07**	**11,24%**	**1,11E+07**	**1,44E+07**
53	155	38	256	512	843964	882482	4,56%	771151	993812
54	4	3	512	512	457746	498283	8,86%	44088,3	952478
55	102	8	256	1024	6,65E+06	6,09E+06	8,36%	5,99E+06	6,19E+06
56	163	38	256	512	1,07E+06	921305	14,27%	805981	1,04E+06
57	79	40	128	256	389058	424844	9,20%	394579	455109
58	**153**	**97**	**512**	**512**	**4,79E+06**	**4,49E+06**	**6,15%**	**3,83E+06**	**5,15E+06**
59	7	2	512	1024	1,77E+06	1,84E+06	3,79%	1,44E+06	2,24E+06
60	229	146	512	1024	1,17E+07	1,09E+07	6,63%	1,02E+07	1,16E+07
61	102	32	256	512	5,08E+06	4,78E+06	5,89%	4,63E+06	4,93E+06
62	37	138	512	256	137316	127979	6,80%	-44221	300178
63	138	155	256	256	796892	744633	6,56%	621251	868015
64	158	175	512	512	4,45E+06	4,26E+06	4,24%	3,49E+06	5,04E+06
65	480	301	1024	1024	1,02E+07	1,07E+07	4,68%	8,53E+06	1,28E+07
66	379	276	256	1024	1,06E+07	9,88E+06	7,11%	9,78E+06	9,98E+06
67	92	115	128	256	977091	1,04E+06	6,25%	1,00E+06	1,08E+06
68	229	153	512	512	1,38E+07	1,32E+07	4,70%	1,23E+07	1,41E+07
69	19	19	256	256	1,94E+06	1,84E+06	4,79%	1,68E+06	2,01E+06
70	**140**	**140**	**1024**	**1024**	**3,99E+07**	**3,81E+07**	**4,47%**	**3,44E+07**	**4,18E+07**
71	9	33	1024	512	6,26E+06	6,02E+06	3,75%	4,30E+06	7,75E+06
72	175	175	512	512	4,44E+06	4,58E+06	3,08%	4,03E+06	5,13E+06
73	129	138	256	256	697272	671954	3,63%	547567	796341
74	37	26	512	512	1,51E+07	1,48E+07	2,44%	1,38E+07	1,57E+07
75	414	414	512	1024	1,48E+07	1,40E+07	5,18%	1,33E+07	1,47E+07
76	139	161	256	256	1,06E+06	1,04E+06	1,73%	898663	1,18E+06
77	5	2	512	1024	1,12E+07	1,12E+07	0,01%	1,04E+07	1,19E+07
78	225	215	256	512	3,79E+06	3,68E+06	2,83%	3,52E+06	3,84E+06
79	**161**	**230**	**128**	**256**	**2,22E+06**	**2,15E+06**	**3,12%**	**2,10E+06**	**2,19E+06**
80	191	188	256	1024	1,70E+06	1,64E+06	3,51%	1,54E+06	1,74E+06
81	8	28	1024	512	1,28E+07	1,27E+07	0,84%	1,08E+07	1,46E+07
82	13	13	1024	1024	2,63E+07	2,72E+07	3,52%	2,41E+07	3,03E+07
83	125	21	256	512	3,13E+06	3,16E+06	1,18%	3,06E+06	3,27E+06
84	263	207	256	128	1,76E+06	1,70E+06	3,47%	1,50E+06	1,90E+06
85	397	397	1024	1024	1,79E+07	1,71E+07	4,53%	1,42E+07	2,00E+07
86	209	216	256	512	3,75E+06	3,65E+06	2,54%	3,53E+06	3,77E+06
87	114	114	512	512	7,05E+06	7,21E+06	2,33%	6,58E+06	7,85E+06
88	79	115	128	256	1,04E+06	1,05E+06	1,35%	1,01E+06	1,09E+06
89	280	209	128	256	936733	940258	0,38%	904778	975739
90	**24**	**14**	**1024**	**1024**	**5,33E+07**	**5,23E+07**	**1,96%**	**4,92E+07**	**5,53E+07**
91	14	14	1024	1024	3,73E+07	3,70E+07	0,87%	3,37E+07	4,02E+07
92	75	32	256	512	1,15E+06	1,18E+06	3,09%	1,08E+06	1,29E+06
93	3	4	512	512	8,35E+06	8,49E+06	1,71%	7,66E+06	9,33E+06
94	125	161	256	256	2,45E+06	2,44E+06	0,42%	2,26E+06	2,61E+06
95	9	9	1024	1024	3,95E+07	3,97E+07	0,36%	3,68E+07	4,25E+07
96	434	419	1024	512	1,46E+07	1,41E+07	3,29%	1,22E+07	1,60E+07
97	**188**	**188**	**1024**	**1024**	**1,24E+08**	**1,24E+08**	**0,24%**	**1,20E+08**	**1,28E+08**
98	129	228	256	128	2,08E+06	2,14E+06	2,67%	1,98E+06	2,29E+06
99	205	146	256	1024	5,54E+06	5,54E+06	0,01%	5,44E+06	5,65E+06
100	207	263	128	256	306469	307855	0,45%	284318	331393

Tabela 8 – Tabela de resultados com erros abaixo de 10% (segunda parte)

	ID1	ID2	AUR	AUA	MIN2	MAX2	%EAUA	SRaster	MIN3	MAX3	SReal	%ES	%DE
51	68	158	3,45E+07	3,44E+07	3,34E+07	3,54E+07	0,15%	8,99%	6,85%	11,25%	8,19%	9,80%	0,80%
52	**188**	**379**	**1,72E+08**	**1,74E+08**	**1,70E+08**	**1,78E+08**	**1,46%**	**7,33%**	**6,23%**	**8,49%**	**6,69%**	**9,64%**	**0,64%**
53	155	38	1,46E+07	1,40E+07	1,31E+07	1,49E+07	4,52%	6,32%	5,19%	7,61%	5,77%	9,52%	0,55%
54	4	3	3,84E+07	3,83E+07	3,71E+07	3,94E+07	0,45%	1,30%	0,11%	2,56%	1,19%	9,35%	0,11%
55	102	8	8,85E+07	8,89E+07	8,54E+07	9,23E+07	0,38%	6,86%	6,50%	7,25%	7,51%	8,70%	0,65%
56	163	38	1,74E+07	1,63E+07	1,53E+07	1,72E+07	6,38%	5,67%	4,69%	6,77%	6,19%	8,43%	0,52%
57	79	40	8,63E+06	8,72E+06	8,48E+06	8,96E+06	1,05%	4,87%	4,41%	5,37%	4,51%	8,06%	0,36%
58	**153**	**97**	**3,35E+07**	**3,41E+07**	**3,31E+07**	**3,52E+07**	**1,97%**	**13,15%**	**10,89%**	**15,56%**	**14,29%**	**7,97%**	**1,14%**
59	7	2	6,99E+07	6,74E+07	6,37E+07	7,10E+07	3,56%	2,73%	2,02%	3,51%	2,53%	7,62%	0,19%
60	229	146	8,68E+07	8,76E+07	8,42E+07	9,09E+07	0,87%	12,49%	11,27%	13,81%	13,50%	7,44%	1,00%
61	102	32	3,45E+07	3,50E+07	3,41E+07	3,59E+07	1,47%	13,66%	12,88%	14,48%	14,73%	7,26%	1,07%
62	37	138	3,11E+07	3,12E+07	3,02E+07	3,22E+07	0,32%	0,41%	-0,14%	1,00%	0,44%	7,10%	0,03%
63	138	155	6,72E+06	6,75E+06	6,50E+06	7,00E+06	0,38%	11,03%	8,88%	13,35%	11,85%	6,91%	0,82%
64	158	175	3,47E+07	3,54E+07	3,44E+07	3,64E+07	2,06%	12,05%	9,58%	14,66%	12,84%	6,17%	0,79%
65	480	301	9,32E+07	9,20E+07	8,82E+07	9,58E+07	1,28%	11,58%	8,91%	14,47%	10,92%	6,04%	0,66%
66	379	276	6,78E+07	6,68E+07	6,32E+07	7,05E+07	1,41%	14,78%	13,87%	15,80%	15,69%	5,79%	0,91%
67	92	115	6,94E+06	6,98E+06	6,76E+06	7,20E+06	0,51%	14,87%	13,89%	15,92%	14,07%	5,70%	0,80%
68	229	153	3,16E+07	3,19E+07	3,09E+07	3,28E+07	0,66%	41,37%	37,54%	45,42%	43,70%	5,33%	2,33%
69	19	19	1,22E+07	1,22E+07	1,20E+07	1,25E+07	0,34%	15,07%	13,44%	16,77%	15,89%	5,11%	0,81%
70	**140**	**140**	**8,10E+07**	**8,15E+07**	**7,84E+07**	**8,47E+07**	**0,59%**	**46,78%**	**40,67%**	**53,38%**	**49,26%**	**5,03%**	**2,48%**
71	9	33	1,36E+08	1,38E+08	1,34E+08	1,42E+08	1,07%	4,37%	3,03%	5,78%	4,59%	4,77%	0,22%
72	175	175	2,84E+07	2,80E+07	2,70E+07	2,90E+07	1,49%	16,36%	13,88%	19,04%	15,64%	4,64%	0,73%
73	129	138	9,88E+06	9,96E+06	9,71E+06	1,02E+07	0,85%	6,75%	5,36%	8,20%	7,06%	4,44%	0,31%
74	37	26	4,56E+07	4,65E+07	4,54E+07	4,75E+07	1,99%	31,76%	29,01%	34,64%	33,21%	4,34%	1,44%
75	414	414	7,30E+07	7,21E+07	6,87E+07	7,55E+07	1,23%	19,46%	17,68%	21,42%	20,27%	4,00%	0,81%
76	139	161	9,18E+06	9,39E+06	9,12E+06	9,66E+06	2,31%	11,05%	9,31%	12,90%	11,51%	3,94%	0,45%
77	5	2	6,26E+07	6,03E+07	5,70E+07	6,36E+07	3,75%	18,50%	16,35%	20,90%	17,81%	3,88%	0,69%
78	225	215	4,65E+07	4,69E+07	4,58E+07	4,80E+07	0,86%	7,86%	7,34%	8,40%	8,16%	3,66%	0,30%
79	**161**	**230**	**8,15E+06**	**8,19E+06**	**7,97E+06**	**8,42E+06**	**0,45%**	**26,20%**	**24,97%**	**27,49%**	**27,16%**	**3,56%**	**0,97%**
80	191	188	1,80E+08	1,80E+08	1,75E+08	1,84E+08	0,28%	0,91%	0,84%	0,99%	0,94%	3,24%	0,03%
81	8	28	7,78E+07	7,94E+07	7,62E+07	8,27E+07	2,05%	16,00%	13,09%	19,16%	16,47%	2,83%	0,47%
82	13	13	2,04E+08	2,05E+08	2,01E+08	2,09E+08	0,78%	13,25%	11,53%	15,03%	12,90%	2,72%	0,35%
83	125	21	3,29E+07	3,24E+07	3,14E+07	3,34E+07	1,34%	9,75%	9,16%	10,38%	9,51%	2,55%	0,24%
84	263	207	6,93E+06	6,86E+06	6,63E+06	7,10E+06	0,97%	24,72%	21,11%	28,59%	25,36%	2,53%	0,64%
85	397	397	8,44E+07	8,26E+07	7,90E+07	8,62E+07	2,18%	20,70%	16,45%	25,35%	21,21%	2,40%	0,51%
86	209	216	2,22E+07	2,22E+07	2,12E+07	2,31E+07	0,28%	16,48%	15,26%	17,80%	16,86%	2,26%	0,38%
87	114	114	4,65E+07	4,67E+07	4,57E+07	4,76E+07	0,42%	15,46%	13,82%	17,17%	15,17%	1,90%	0,29%
88	79	115	4,74E+06	4,88E+06	4,67E+06	5,10E+06	2,96%	21,53%	19,87%	23,34%	21,87%	1,56%	0,34%
89	280	209	7,49E+06	7,63E+06	7,43E+06	7,84E+06	1,97%	12,32%	11,54%	13,13%	12,51%	1,56%	0,20%
90	**24**	**14**	**1,42E+08**	**1,41E+08**	**1,38E+08**	**1,45E+08**	**0,45%**	**36,98%**	**33,94%**	**40,18%**	**37,55%**	**1,51%**	**0,57%**
91	14	14	2,21E+08	2,22E+08	2,17E+08	2,26E+08	0,31%	16,70%	14,96%	18,50%	16,90%	1,18%	0,20%
92	75	32	2,72E+07	2,77E+07	2,67E+07	2,87E+07	1,91%	4,26%	3,75%	4,82%	4,21%	1,15%	0,05%
93	3	4	3,06E+07	3,14E+07	3,05E+07	3,23E+07	2,73%	27,05%	23,68%	30,63%	27,32%	1,00%	0,27%
94	125	161	1,02E+07	1,03E+07	1,00E+07	1,05E+07	0,34%	23,71%	21,53%	25,99%	23,89%	0,75%	0,18%
95	9	9	1,87E+08	1,89E+08	1,85E+08	1,93E+08	0,88%	21,02%	19,09%	23,03%	21,13%	0,52%	0,11%
96	434	419	9,08E+07	8,81E+07	8,47E+07	9,15E+07	2,99%	16,00%	13,35%	18,86%	16,05%	0,30%	0,05%
97	**188**	**188**	**2,17E+08**	**2,18E+08**	**2,14E+08**	**2,22E+08**	**0,51%**	**56,88%**	**54,01%**	**59,85%**	**57,03%**	**0,28%**	**0,16%**
98	129	228	6,63E+06	6,80E+06	6,60E+06	7,00E+06	2,54%	31,42%	28,34%	34,70%	31,38%	0,13%	0,04%
99	205	146	7,42E+07	7,42E+07	7,09E+07	7,75E+07	0,00%	7,47%	7,02%	7,97%	7,47%	0,01%	0,00%
100	207	263	8,38E+06	8,42E+06	8,16E+06	8,68E+06	0,45%	3,66%	3,27%	4,06%	3,66%	0,01%	0,00%

5.3 Análise de resultados

Como o cálculo da similaridade raster é feito pela razão das áreas aproximadas de interseção e de união, é importante analisar quanto estes valores influenciam na precisão do resultado. A Figura 25 apresenta a relação entre os percentuais de erro da área de interseção, de erro da área de união e de erro da similaridade raster. O eixo Y apresenta os percentuais de erro, enquanto que o eixo X apresenta os objetos ordenados do maior erro para o menor erro. Observe que o erro da área de união é

relativamente baixo, enquanto que o valor do erro da similaridade raster acompanha o valor do erro da área de interseção. Portanto, quanto menor o erro da área de interseção, menor o erro da similaridade raster.

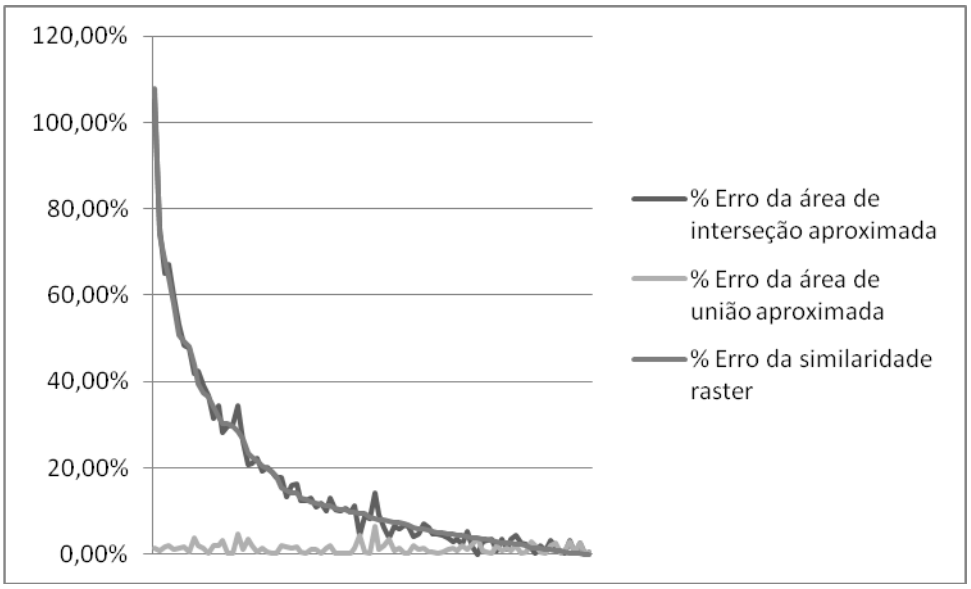

Figura 25 – Percentuais de erro da área de interseção, área de união e similaridade

Analisando o pior resultado do algoritmo, destacado na linha 1 da Tabela 5 e da Tabela 6 e representado pelos objetos apresentados na Figura 26, podemos observar que a interseção entre os objetos é muito pequena. Somente duas células têm interseção, como apresentado na coluna *NC*. Neste caso, a similaridade raster é de 0,58%, no entanto a similaridade real é aproximadamente igual a 0% (Tabela 6). Logo, o erro na estimativa de similaridade (%ES) utilizando tipos de célula foi muito grande (257.252,94%). Conseqüentemente, usar a aproximação para estimar a similaridade neste caso se torna muito ruim. Portanto, quando a similaridade é muito pequena, pode ser mais vantajoso realizar a consulta exata.

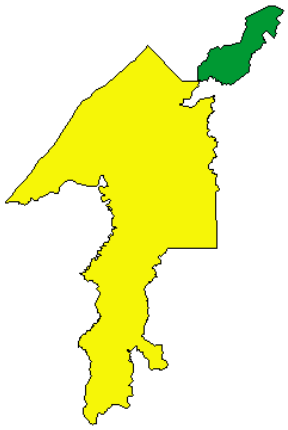

Figura 26 – Objeto 234 x Objeto 397

O mesmo acontece no segundo caso, destacado na linha 2 da Tabela 5 e da Tabela 6 e representado pelos objetos apresentados na Figura 27, onde o erro na estimativa é de 217.074,82%. A similaridade raster é igual a 2,58%, enquanto que a simila-

ridade real é próxima de 0% (Tabela 6). Neste caso, existem somente quatro células com interseção.

Figura 27 – Objeto 10 x Objeto 25

É interessante notar também que nos dois exemplos ocorre uma mudança de escala de 2^8 para 2^{10}, ocasionando uma perda de informação. Como apresentado na seção 2.1.1, a mudança de escala é feita agrupando-se um conjunto de células de tamanho menor para representar uma célula de tamanho maior somando pesos para cada tipo de célula de maneira pessimista. Isto leva a maior perda na precisão do algoritmo.

Analisando outro resultado ruim, destacado na linha 10 das Tabela 5 e Tabela 6 e representado pelos objetos da Figura 28, o percentual de erro (%ES) foi de 108,86%. A similaridade raster é igual a 10,93% e a similaridade real é igual a 7,48% (Tabela 6). Neste caso, podemos verificar que o número de células com interseção (*NC*) é um pouco maior (vinte e seis células). Entretanto, este número continua pequeno, e é agravado pelo fato de que houve uma mudança de escala do fator de 2^9 para 2^{10}, perdendo mais informação e aumentando o erro da aproximação. Analisando a quantidade de células que têm interseção de acordo com seus tipos (Pouco × Pouco, Pouco × Muito, Pouco × Cheio, Muito × Muito, Muito × Cheio e Cheio × Cheio) (Tabela 9), observamos que não existem interseções de células do tipo *Cheio* × *Cheio*, cuja interseção tem 100% de precisão e que, caso ocorressem, poderiam levar a uma maior precisão.

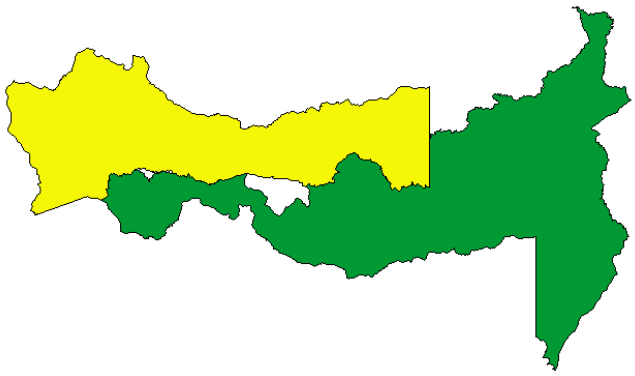

Figura 28 – Objeto 2 x Objeto 5

Tabela 9 - Número de células que têm interseção - Objeto 2 x Objeto 5

P × P	P × M	P × C	M × M	M × C	C × C
2	7	6	6	5	0

Por outro lado, analisando o caso destacado na linha 11 da Tabela 5 e da Tabela 6 e representado pelos objetos da Figura 29, podemos observar que não há mudança de escala, mas, em compensação, o número de células que têm interseção é muito pequeno (nove células) e a similaridade raster e a similaridade real também são muito pequenas (0,98% e 0,46%, respectivamente - Tabela 6). Neste caso, a diferença entre a similaridade calculada através das assinaturas 4CRS dos objetos e a similaridade calculada a partir dos valores exatos também foi grande (107,88%).

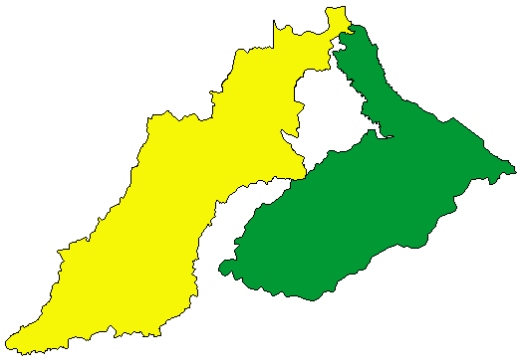

Figura 29 – Objeto 140 x Objeto 146

Em outros dois casos, destacados, respectivamente, nas linhas 16 e 17, da Tabela 5 e da Tabela 6 e representados pelos objetos da Figura 30 e da Figura 31, o erro na estimativa é de 50,65% e 49,36%, respectivamente. Podemos observar que o número de células que têm interseção é muito pequeno (dez células). Analisando a quantidade de células que têm interseção de acordo com seus tipos (Tabela 10 e Tabela 11), observamos que não existem interseções de células do tipo *Cheio* × *Cheio*, cuja interseção tem 100% de precisão.

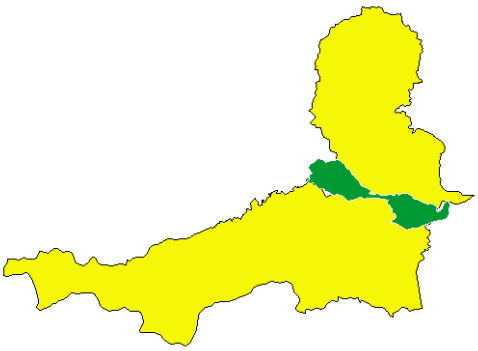

Figura 30 – Objeto 183 x Objeto 44

Tabela 10 - Número de células que têm interseção - Objeto 183 x Objeto 5

P × P	P × M	P × C	M × M	M × C	C × C
3	1	4	2	0	0

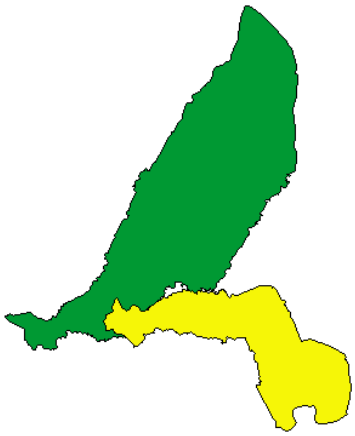

Figura 31 – Objeto 3 x Objeto 15

Tabela 11 - Número de células que têm interseção - Objeto 3 x Objeto 15

P × P	P × M	P × C	M × M	M × C	C × C
4	3	1	2	0	0

Podemos observar em todos os casos apresentados (Figura 26 a Figura 30) que são três os principais fatores que influenciaram no resultado:

- Número pequeno de células com interseção entre as assinaturas e, conseqüentemente, similaridade com valor muito baixo;
- Maioria das células com interseção é do tipo onde é feita uma aproximação considerando a média (*Pouco* × *Pouco*, *Pouco* × *Muito*, *Pouco* × *Cheio*, *Muito* × *Muito* ou *Muito* × *Cheio*);
- Mudança de escala para comparação.

Por outro lado, o algoritmo retorna bons resultados em casos onde estes fatores são minimizados. Por exemplo, no caso destacado na linha 52 da Tabela 7 e da Tabela 8 e representado pelos objetos da Figura 32, onde o erro na estimativa é de 9,64%. A similaridade raster é igual a 7,33% e a similaridade real é igual a 6,69%. Neste caso, o número de células com interseção também é pequeno (vinte e duas células), porém, agora existem interseções do tipo *Cheio* × *Cheio,* onde a precisão é de 100%, como apresentado na Tabela 12.

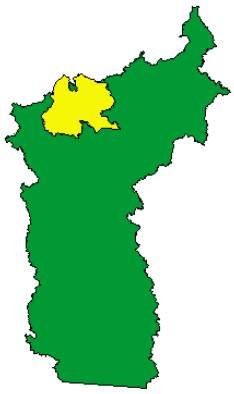

Figura 32 – Objeto 188 x Objeto 379

Tabela 12 – Número de células que têm interseção - Objeto 188 x Objeto 379

P × P	P × M	P × C	M × M	M × C	C × C
2	1	6	3	6	4

Analisando outro caso, com resultado ainda melhor, destacado na linha 58 da Tabela 7 e da Tabela 8 e representado pelos objetos da Figura 33, o erro na estimativa é de 7,97%, a similaridade raster é igual a 13,15% e a similaridade real é igual a 14,29%. Neste caso, observamos que existem células com interseção do tipo *Cheio* × *Cheio* e que o número total de células com interseção é maior (trinta e cinco células).

Figura 33 – Objeto 153 x Objeto 97

Tabela 13 - Número de células que têm interseção - Objeto 153 x Objeto 97

P × P	P × M	P × C	M × M	M × C	C × C
3	6	8	2	5	2

Agora, um exemplo bem interessante, onde um objeto é comparado com ele mesmo deslocado, destacado na linha 70 da Tabela 7 e da Tabela 8 e representado pelos objetos da Figura 34, onde o erro na estimativa é de 5,03%, a similaridade raster é igual a 46,78% e a similaridade real é igual a 49,26%. Neste caso, o número de células com interseção é bem maior que nos exemplos anteriores (sessenta e sete células) e boa parte dessas interseções de células é do tipo *Cheio* × *Cheio,* como apresentado na Tabela 14.

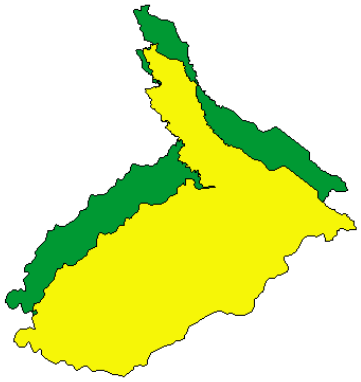

Figura 34 – Objeto140 x Objeto 140

Tabela 14 - Número de células que têm interseção - Objeto 140 x Objeto 140

P × P	P × M	P × C	M × M	M × C	C × C
5	10	15	6	16	15

Em outro exemplo, destacado na linha 79 da Tabela 7 e da Tabela 8 e representado pelos objetos da Figura 35, o erro na estimativa é de 3,56%, a similaridade raster é igual a 26,20% e a similaridade real é igual a 27,16%. Neste caso, o número de células com interseção também é grande (sessenta e sete células) e boa parte dessas células é do tipo *Cheio* × *Cheio,* como apresentado na Tabela 15.

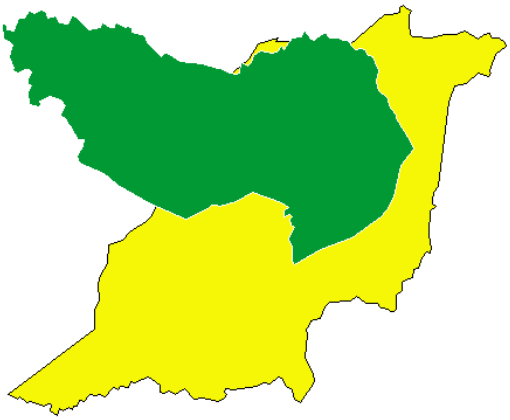

Figura 35 – Objeto 161 x Objeto 230

Tabela 15 - Número de células que têm interseção - Objeto 161 x Objeto 230

P × P	P × M	P × C	M × M	M × C	C × C
4	2	15	3	13	17

Outro resultado muito bom é destacado na linha 90 da Tabela 7 e da Tabela 8 e representado pelos objetos da Figura 36. O erro na estimativa é de 1,51%, a similaridade raster é igual a 36,98% e a similaridade real é igual a 37,55%. Neste caso, o número de células com interseção é bem grande (oitenta células) e, mais uma vez, boa parte dessas células é do tipo *Cheio* × *Cheio,* como apresentado na Tabela 16.

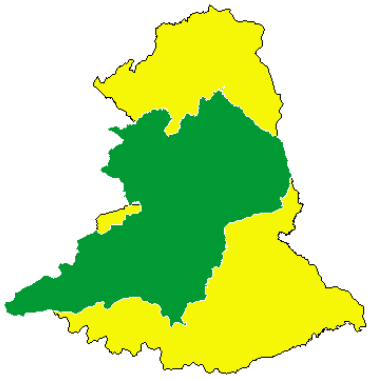

Figura 36 – Objeto 24 x Objeto 14

Tabela 16 - Número de células que têm interseção - Objeto 24 x Objeto 14

P × P	P × M	P × C	M × M	M × C	C × C
5	7	18	5	16	29

Um dos melhores resultados, onde mais uma vez, um objeto é comparado com ele mesmo deslocado, é destacado na linha 97 das Tabela 7 e Tabela 8, e representado pelos objetos da Figura 37. O erro de estimativa nesse caso é de apenas 0,28%, a similaridade raster é igual a 56,88% e a similaridade real é igual a 57,03%. Neste exemplo, o número de células com interseção é ainda maior que no exemplo anterior (cento e sessenta e três células) e boa parte dessas interseções de células é do tipo *Cheio* × *Cheio,* como apresentado na Tabela 17.

Figura 37 – Objeto 188 x Objeto 188

Tabela 17 - Número de células que têm interseção - Objeto 188 x Objeto 188

P × P	P × M	P × C	M × M	M × C	C × C
10	9	23	7	33	81

A imprevisibilidade do algoritmo, em muitos casos, pode ser explicada pela aproximação da área de interseção entre células dos tipos *Pouco* × *Pouco*, *Pouco* × *Muito*, *Pouco* × *Cheio*, *Muito* × *Muito* e *Muito* × *Cheio.* Quando este número é muito pequeno, não podemos supor a distribuição normal. Logo, dois casos podem ocorrer.

No primeiro, quando a área de interseção entre a maioria das células se aproxima da média, o resultado aproximado será bem próximo do real. Porém, no segundo caso, quando esta área se distancia da média, o resultado aproximado será bem distante do resultado exato.

Pudemos notar que todos os exemplos destacados nos resultados com erros acima de 10%, tinham como interseção a borda dos objetos, enquanto que em todos os exemplos destacados nos resultados com erros abaixo de 10%, os objetos tinham boa parte de suas áreas sobrepostas.

Logo, temos fortes indícios de que o que determinou a precisão do algoritmo foi a aproximação da área de interseção entre células dos tipos *Pouco* × *Pouco*, *Pouco* × *Muito, Muito* × *Muito* e *Muito* × *Cheio.* Dessa forma, é necessário um estudo para melhorar o algoritmo que calcula a área de interseção a partir de assinaturas raster.

No segundo caso, onde temos um percentual grande de sobreposição dos objetos, os resultados foram satisfatórios. Dessa forma, quanto maior o valor da similaridade raster, mais próximo ele está da similaridade real. Por outro lado, existe um erro percentual grande quando o valor da similaridade raster é pequeno. Para estes casos, indicamos o uso do cálculo exato.

Quanto ao intervalo de confiança calculado, para os resultados com erros acima de 10%, em 70% dos casos, o valor da similaridade real (calculada a partir dos objetos reais) ficou dentro do intervalo. Em contrapartida, para os resultados com erros abaixo de 10%, em 96% dos casos, o valor da similaridade real ficou dentro do intervalo. Este intervalo pode ser consultado através dos campos *MIN3* e *MAX3* da Tabela 6 (resultados com erros acima de 10%) e da Tabela 8 (resultados com erros abaixo de 10%).

6 Conclusão

A similaridade entre objetos é um conceito fundamental. Uma métrica de similaridade fornece uma medida de semelhança entre pares de coisas que permite identificar a que classes elas pertencem. Existem várias áreas de aplicação para uso da similaridade, como apontado por Sako e Fujimura [2000]: bancos de dados de imagens médicas, reconhecimento de gestos/movimentos humanos, sistemas de informações geológicas, comércio eletrônico, proteção de direitos autorais, design gráfico, arte criativa, dentre outras.

Este trabalho tem como principal contribuição a proposta de um algoritmo para calcular a similaridade entre polígonos a partir de suas assinaturas 4CRS. Outras contribuições são a proposta de algoritmo para calcular a união de duas assinaturas 4CRS e a implementação no Secondo [Güting *et al.*, 2005] dos algoritmos propostos e dos algoritmos para calcular a área aproximada de um polígono e a área de interseção aproximada entre dois polígonos, sendo os dois últimos propostos por Azevedo *et al.* [2004] e Azevedo *et al* [2005].

Testes experimentais foram realizados sobre dados reais de municípios do norte do Brasil a fim de avaliar a precisão do algoritmo. Os resultados obtidos demonstraram que existe uma variação da precisão do algoritmo, que foram justificadas através dos exemplos apresentados.

Os casos em que há um erro superior a 10% têm como principais características: número pequeno de células com interseção entre as assinaturas e, conseqüentemente, similaridade com valor muito baixo; maioria das células com interseção é do tipo onde é feita uma aproximação considerando a média (*Pouco* × *Pouco*, *Pouco* × *Muito*, *Pouco* × *Cheio*, *Muito* × *Muito* ou *Muito* × *Cheio*), ou seja, a interseção ocorre nas bordas dos objetos;e, ocorreu mudança de escala para executar os algoritmos de interseção aproximada e união de assinaturas raster.

Os resultados para casos onde temos um percentual grande de sobreposição dos objetos foram satisfatórios. Logo, podemos afirmar que, quanto maior o valor da similaridade raster, mais próximo ele está da similaridade real. Por outro lado, existe um erro percentual grande quando o valor da similaridade raster é pequeno. Logo, se a aplicação do algoritmo está buscando encontrar objetos com similaridade alta, indicamos que assinaturas com similaridade baixa não devem ser consideradas

no resultado. Por outro lado, se há interesse em ter uma noção de similaridade baixa ou alta, então para os casos de similaridade baixa, indicamos que o cálculo exato deve ser realizado.

Em relação à fórmula para cálculo do intervalo de confiança, o valor exato ficou dentro do intervalo de confiança em 96% dos casos, onde o erro na estimativa da similaridade raster era menor que 10%, o que demonstra que para estes casos a similaridade raster tem um bom resultado. Por outro lado, para os casos em que o erro da similaridade raster em relação à similaridade real foi superior a 10%, em 70% dos casos o valor da similaridade real ficou dentro do intervalo de confiança da similaridade raster. Apesar do valor de 70% ser razoável, consideramos que é necessário analisar melhor o algoritmo para melhorá-lo.

Como trabalhos futuros, propomos:

- Melhorar o algoritmo para calcular a área aproximada de interseção, pois há uma relação direta entre o erro resultante deste algoritmo em relação ao algoritmo que calcula a similaridade raster, como demonstrado no gráfico apresentado na Figura 25. Uma possível direção para melhoria deste algoritmo é analisar outras estimativas para a área de interseção entre tipos de células.
- Realizar testes de desempenho comparando o algoritmo de similaridade raster em relação ao cálculo da similaridade real.
- Fazer uma análise detalhada de complexidade dos algoritmos propostos.
- Pesquisar por outros algoritmos que calculam a área aproximada de polígonos e analisá-los em relação ao algoritmo proposto neste trabalho.
- Analisar em detalhes a influência da mudança de escala das células em relação à precisão do resultado.
- Evoluir o algoritmo para calcular a similaridade de objetos sem considerar a posição absoluta. Em outras palavras, mesmo que os objetos não estejam na mesma posição no espaço, calcular a similaridade entre eles. Para realizar este cálculo é necessário definir uma forma normalizada para representar os objetos, por exemplo, deslocar as assinaturas para uma posição no espaço que seja possível realizar a comparação de células, por exemplo, a partir de uma célula que corresponda ao centro dos objetos ou para um ponto na extremidade inferior-esquerda.
- Evoluir o algoritmo para poder ser utilizado em outras situações como, por exemplo, o algoritmo permitir comparar objetos de acordo com sua forma, independente do tamanho dos mesmos e sem a necessidade de mudança de escala. Exemplos de uso deste algoritmo seriam: comparar uma maquete de um objeto (em tamanho reduzido) com um objeto original (em tamanho maior); comparar padrões de imagens médicas.
- Evoluir a assinatura raster para armazenar faces, ou seja, permitir que um polígono seja composto por várias faces. Dessa forma pretende-se aumentar o nível de representação dos objetos e permitir o armazenamento de uma assinatura de união entre duas assinaturas que não tenham interseção uma com a outra.

- Definir novo algoritmo para criar assinatura a fim de diminuir a quantidade de células adjacentes que têm a mesma representação. Uma possibilidade é armazenar em uma única célula a representação de um conjunto de células adjacentes iguais. Outra proposta seria utilizar estrutura hierárquica para armazenar a assinatura, por exemplo, uma QuadTree onde diferentes níveis representariam a assinatura em diferentes escalas. Esta proposta também poderia ser utilizada para calcular a similaridade com diferentes níveis de precisão, ou seja, cada comparação de assinaturas em determinado nível resultará em uma similaridade com determinada precisão.
- Avaliar o uso de algoritmos de compactação para reduzir o espaço utilizado para armazenamento das assinaturas.
- Analisar formas para calcular o intervalo de confiança de acordo com tipos de objetos. Definir características dos objetos que indiquem se objeto resultará em cálculos com intervalo de confiança mais preciso. Por exemplo, objetos que não são convexos podem levar a um resultado de similaridade com maior erro do que objetos convexos.
- Analisar possibilidade de acelerar consultas em grandes volumes de dados aproximados sem ter que comparar objeto a objeto. Técnicas de indexação poderiam ser estudadas a fim de considerar assinaturas 4CRS em múltiplos níveis. Dessa forma, ao comparar a assinatura de um objeto com a assinatura de um nível e o resultado for ruim, não se seguiria no cálculo da similaridade com os níveis mais baixos.
- Implementar a visualização de assinaturas Raster no Secondo. A visualização das assinaturas pode auxiliar muito nas análises dos resultados obtidos.

Referências Bibliográficas

ARONOFF, S., **Geographic Information Systems**. 1 ed. Ottawa, Canada, WDL Publications, 1989.

AZEVEDO, L. G., MONTEIRO, R. S., ZIMBRÃO, G.; SOUZA, J. M., **Approximate Spatial Query Processing Using Raster Signatures**. In: VI Simpósio Brasileiro de GeoInformática (GeoInfo 2004), Campos do Jordão, Brasil, 2004.

AZEVEDO, L. G., ZIMBRÃO, G., SOUZA, J. M.; GÜTING, R. H., **Estimating the Overlapping Area of Polygon Join**. In: International Symposium on Advances in Spatial and Temporal Databases, v. 1. p. 91-108, Angra dos Reis, Brasil, 2005.

AZEVEDO, L. G. **Processamento Aproximado de Consultas em Bancos de Dados Espaciais Usando Assinaturas Raster**. Tese de Doutorado do Programa de Pós-Graduação de Engenharia da Universidade Federal do Rio de Janeiro. 120 pg., 2005.

AZEVEDO, L. G., ZIMBRÃO, G., SOUZA, J. M., **Approximate Query Processing in Spatial Databases Using Raster Signatures**. In: Davis, Clodoveu A.D. Jr.; Monteiro,

Antonio M.V.M.. (Org.). Advances in Geoinformatics. 1 ed. Heidelberg: Springer, 2006, v. 1, p. 69-85.

BRINKHOFF T., KRIEGEL H.-P., SCHNEIDER R., **Comparison of Approximations of Complex Objects used for Approximation-based Query Processing in Spatial Database Systems.** Proc. 9th Int. Conf. on Data Engineering, Vienna, Austria, 1993, pp. 40-49.

BRINKHOFF, T., KRIEGEL, H. P., SCHNEIDER, R., SEEGER, B., 1994, **Multi-step Processing of Spatial Joins**. *ACM SIGMOD Record*, v. 23, n.2 (Jun), pp. 197-208.

CAKMAKOV, D., CELAKOSKA, E., **Estimation of Curve Similarity Using Turning Function.** International Journal of Applied Math., vol. 15, no. 4, pp. 403-416, 2004.

DOBRA, A., GAROFALAKIS, M., GEHRKE, J. E., RASTOGI, R., **Processing complex aggregate queries over data streams**. In: Proceedings of the 2002 ACM-SIGMOD International Conference on Management of Data, pp. 61-72, Madison, Wisconsin, USA, Jun. 2002.

GIBBONS, P. B., MATIAS, Y., POOSALA, V., **Aqua project white paper**. Technical Report, Bell Laboratories, Murray Hill, New Jersey, USA, 1997.

GOODMAN N., **Seven Strictures on Similarity**, In: Problems and Projects, Bobbs-Merrill, Indianapolis, 437-446, 1972.

GÜTING, R. H., **An Introduction to Spatial Database Systems**. The International Journal on Very Large Data Bases, v. 3, n. 4 (Oct), pp. 357-399, 1994.

GÜTING, R. H., ALMEIDA, V., ANSORGE, D., BEHR T., DING, Z., HÖSE, T., HOFFMANN F., SPIEKERMANN, M., **SECONDO: An Extensible DBMS Platform for Research Prototyping and Teaching**. Demo-Paper, 21st Intl. Conf. on Data Engineering (ICDE, Tokyo, Japan), 1115-1116, 2005.

GÜTING, R., H., ANSORGE, D., DÜNTGEN, C., JANDT, S., BEHR, T., SPIEKERMANN, M., **SECONDO 3.0 User Manual.** Version 4.3, June 16, 2010. Hagen, Germany. <http://dna.fernuni-hagen.de/Secondo.html/>

JACCARD, P. The distribution of flora in the alpine zone. The New Phytologist, vol. 11(2), pp. 37–50, 1912.

HEMERT, J. V., BALDOCK, R., **Mining Spatial Gene Expression Data for Association Rules**, BIRD 2007, LNBI 4414, pp 66-76, Springer, 2007.

HOLT, A. (2003) **Spatial similarity**. In: 15th Annual Colloquium of the Spatial Information Research Centre (SIRC 2003: Land, Place and Space), 1-2 December 2003, Dunedin, New Zealand, pp. 77-80.

QUINE, W. V., **Ontological Relativity and Other Essays**, Columbia University Press, New York, 1969.

SAKO Y., FUJIMURA K., **Shape Similarity by Homotopic Deformation**. *The Visual Computer*, *16*(1), 2000, 47-61.

SAMET, H., **The Design and Analysis of Spatial Data Structure**. Addison-Wesley Publishing Company, 1a ed., Boston, Massachusetts, 1990.

SCHNEIDER R.: **A Storage and Access Structure for Spatial Database Systems** (in German), Ph.D. thesis, Institute for Computer Science, University of Munich, 1992.

TAO, Y., SUN, J., PAPADIAS, D., **Selectivity estimation for predictive spatio-temporal queries**. In: Proceedings of the 19th International Conference on Data Engineering, pp. 417-428, Bangalore, India, 2003.

THEODORIDIS, Y. SELLIS, T.K. PAPADOPOULOS, A.N. MANOLOPOULOS, Y., **Specifications for efficient indexing in spatiotemporal databases**. In: Proceedings of the tenth International Conference on Scientific and Statistical Database Management, p. 123-132, Jul. 1998.

TVERSKY A., **Features of Similarity**, Psychological Review, 84(4), 327-352, 1977.

YANCHI L., HONGWEI G., XUEDONG G., **Analysis of Blast Furnace Cross Thermometric Based on Spatial Data Mining,** International Conference onCyber-Enabled Distributed Computing and Knowledge Discovery (Cyber-C), p.33-36, Oct. 2009.

ZIMBRAO, G., SOUZA, J. M., "**A Raster Approximation for Processing of Spatial Joins**". In: Proceedings of the 24rd International Conference on Very Large Data Bases, pp. 558-569, New York City, New York, USA, Aug. 1998.

Apêndice 1 Configuração necessária para incluir a álgebra Raster no secondo

Para vincular uma álgebra ao SECONDO, ela deve ser registrada no gerenciador de álgebras e instalada no processo de compilação (*build process*), ou seja, ela tem que ser inserida em um arquivo de configuração e em um *makefile*.

A primeira coisa a fazer é criar um novo diretório no diretório *Algebras* do SECONDO. Por convenção, o nome do novo diretório deve ser o mesmo que o nome da Álgebra, omitindo o sufixo "Algebra" (Raster, neste caso). Depois, criam-se os arquivos da álgebra neste diretório.

Uma álgebra é composta basicamente pelos seguintes arquivos:

- <Algebra>.h: neste arquivo são definidas as classes da álgebra. Estas classes correspondem à representação em memória dos construtores de tipos da álgebra.
- <Algebra>.cpp: neste arquivo estão as implementações das classes existentes em <Algebra>.h.
- <Algebra>.spec: contém as especificações de como utilizar os operadores da álgebra.
- makefile: define o processo de compilação da álgebra.

O arquivo *makefile* inclui informações sobre quais arquivos devem ser compilados durante o processo de compilação do sistema. O caminho mais fácil para criar o

makefile é copiar um *makefile* de outra álgebra e adaptá-lo à nova álgebra. Na maioria dos casos você não precisa fazer mais nada, uma vez que existem regras padronizadas para criação de arquivos .o e arquivos .cpp.

Em seguida, devemos modificar o arquivo *makefile.algebras* (localizado no diretório principal do SECONDO). Este arquivo contém dois registros para cada álgebra. O primeiro define o nome do diretório e o segundo o nome do módulo da álgebra, como apresentado na Figura 38.

```
...
ALGEBRA_DIRS += Raster
ALGEBRAS += RasterAlgebra
ALGEBRA_DIRS += BTree
ALGEBRAS += BTreeAlgebra
...
```

Figura 38 - Inclusão de álgebra no arquivo *makefile.algebras*

O último passo é registrar a álgebra no gerenciador de álgebras. Para isto, ela deve ser adicionada à lista de álgebras em *AlgebraList.i.cfg*, no diretório *Algebras/Management*. Neste arquivo deve ser incluída uma nova linha para a função de inicialização, incluindo como parâmetros deste método o nome da álgebra e um número de álgebra único (apenas escolha um número não utilizado).

```
...
ALGEBRA_INCLUDE(1, StandardAlgebra)
ALGEBRA_INCLUDE(2, FunctionAlgebra)
ALGEBRA_INCLUDE(3, RelationAlgebra)
ALGEBRA_INCLUDE(4, RasterAlgebra)
ALGEBRA_INCLUDE(5, StreamExampleAlgebra)
...
```

Figura 39 - Exemplo de include para álgebra no arquivo *AlgebraList.i.cfg*

Agora, todas as configurações estão concluídas para a nova Álgebra. Execute o comando *make* no MSYS no diretório principal do SECONDO para vincular a nova álgebra. Depois que todo o processo for finalizado, inicie uma interface com o usuário do SECONDO (por exemplo, SecondoTTYBDB) e digite *list algebra <nome da álgebra>Algebra* (por exemplo, *list algebra RasterAlgebra*) para visualizar seus operadores e construtores de tipo.

É importante observar que, caso deseje-se fazer uma compilação mais rápida do SECONDO, compilando apenas as interfaces monousuário, pode-se executar o comando *make TTY*, ao invés do comando *make*. Neste caso, são compilados apenas os módulos TTY (por exemplo, SecondoTTYBDB) e não, por exemplo, a interface gráfica JavaGui. Usar esta opção de compilação é uma boa abordagem quando se deseja compilar rapidamente as alterações realizadas utilizando apenas a interface de linha de comando.

Apêndice 2 Criação do operador RSimilar na álgebra Raster

A implementação de um operador no SECONDO deve incluir os seguintes itens:

- Uma função de mapeamento de tipo;
- Uma função de seleção;
- Uma função de mapeamento de valor;
- Uma descrição do operador; e
- Uma especificação da sintaxe com um exemplo de uso.

A função de mapeamento de tipo verifica se os tipos de argumento estão corretos ou não. Caso não estejam corretos, o resultado é uma lista de expressões com o símbolo *typeerror*, que fará com que os próximos mapeamentos de tipo também falhem. Um exemplo de uma função deste tipo é mostrado na Figura 40.

```
ListExpr
rSimilarTypeMap( ListExpr args )
{
      if ( nl->ListLength(args) != 2 )
      {
          cmsg.typeError("Type mapping function got a parameter of
length != 2.");
        return nl->TypeError();
      }

      ListExpr arg1 = nl->First(args);
      ListExpr arg2 = nl->Second(args);

      if ( nl->IsEqual(arg1, "raster4CRS") && nl->IsEqual(arg2,
"raster4CRS") )
            return nl->SymbolAtom("approxresult");

      return nl->TypeError();
}
```

Figura 40 – Exemplo de função de mapeamento de tipo

A função de seleção é utilizada para selecionar uma dentre as diversas funções para um operador sobrecarregado, com base nos tipos de argumentos. O retorno desta função é o índice de um *array* de funções de mapeamento de valor. Este índice do *array* possui a função adequada. Neste operador, faremos uso da função de seleção trivial, pois não há sobrecarga, ou seja, não há necessidade de seleção de uma função dentre várias opções, como apresentado na Figura 41.

```
Operator::SimpleSelect        //trivial selection function
```

Figura 41 – Exemplo de função de mapeamento de valor

A função de mapeamento de valor é responsável por calcular o valor de resultado de uma operação. Para cada combinação permitida de tipos de argumento, é necessária uma função de mapeamento de valor, que são armazenadas em um *array*. O valor resultante é diretamente escrito em um objeto fornecido pelo processador de consultas. A Figura 42 apresenta um exemplo de uma função de mapeamento de valor do operador *rSimilar*.

```
Int
rSimilarFun( Word* args, Word& result, int message,
                  Word& local, Supplier s )
{
      Raster4CRS* r1 = ((Raster4CRS*)args[0].addr);
      Raster4CRS* r2 = ((Raster4CRS*)args[1].addr);

      result = qp->ResultStorage(s);

      double resultMin = 0, resultMax = 0;
      double similar = r1->similar(r2, resultMin, resultMax);

      ApproxResult*   similarResult   =   new   ApproxResult(similar,
resultMin, resultMax);
      result.addr = similarResult;

      return 0;
}
```

Figura 42 – Exemplo de função de mapeamento de valor

A descrição do operador é o que será exibido quando o comando *list operators* for executado. Um exemplo dessa descrição é apresentado na Figura 43.

```
const   string   rSimilarSpec     =   "(   (   \"Signature\"   \"Syntax\"
\"Meaning\" " "\"Example\" ) "
            "( <text>(Raster4CRS, Raster4CRS) -> real</text--->"
            "<text>_ rSimilar _</text--->"
            "<text>Returns the percentual of similarity between two
raster signatures.</text--->"
            "<text>query  raster4CRS1  rSimilar  raster4CRS2</text---
>"
            ") )";
```

Figura 43 – Exemplo de descrição de operador

Uma definição do operador também deve ser feita, passando como parâmetro todas as funções criadas anteriormente, como é apresentado na Figura 44.

```
Operator RasterSimilar(
  "rSimilar",                          //name
  rSimilarSpec,                        //specification
  rSimilarFun,                         //value mapping
  Operator::SimpleSelect,              //trivial selection function
  rSimilarTypeMap                      //type mapping
);
```

Figura 44 – Exemplo de definição de operador

Por último, o operador deve ser adicionado ao construtor da classe, como ilustrado na Figura 45.

```
class RasterAlgebra : public Algebra
{
 public:
  RasterAlgebra() : Algebra()
  {
      AddTypeConstructor( &TCRaster4CRS );
      TCRaster4CRS.AssociateKind("DATA");

      AddTypeConstructor( &approxresultTC );
      approxresultTC.AssociateKind("DATA");

      AddOperator( Calc4CRSInfo(), Calc4CRSFun,
```

```
                    RegionLinePointsRaster4CRS );
      AddOperator( &Calc3CRS );
      AddOperator( &RasterIntersects );
      AddOperator( &RasterSimilar );                    //Added
      AddOperator( &RasterArea );
      AddOperator( &RasterPlus );
      AddOperator( &RasterIntersection );
      AddOperator( &RasterOverlappingArea );
  }
  ~RasterAlgebra() {};
};
```

Figura 45 – Exemplo de operador adicionado ao construtor da classe

Apêndice 3 Código-fonte dos algoritmos implementados

```
long double Signature4CRS::approximateArea()
{
	double nCellType[4]={0, 0, 0, 0};
	long cellSize = 1l << map->potency;
	MBR alignedMBR = BBox2D::alignedMBR(map->mbr, cellSize);
	return approximateArea(nCellType, alignedMBR);
}

long double Signature4CRS::approximateArea(double &deltaConfidence)
{
	double nCellType[4]={0, 0, 0, 0};
	long cellSize = 1l << map->potency;
	MBR alignedMBR = BBox2D::alignedMBR(map->mbr, cellSize);
	double area = approximateArea(nCellType, alignedMBR,
deltaConfidence);

	double resultMin = area - deltaConfidence;
	double resultMax = area + deltaConfidence;

	return area;
}

long double Signature4CRS::approximateArea(double nCellType[4], MBR
alignedMBR, double &deltaConfidence)
{
	long blockArea = sizeOfBlock * sizeOfBlock;
	double area = approximateArea(nCellType, alignedMBR);
	if ((nCellType[1] != 0) && (nCellType[2] != 0))
		deltaConfidence = ( nCellType[1] * CONFIDENCE_INTERVAL *
sqrt(VARIANCE/nCellType[1]) +
				nCellType[2] * CONFIDENCE_INTERVAL *
sqrt(VARIANCE/nCellType[2]) ) * blockArea;
	return area;
}

long double Signature4CRS::approximateArea(double nCellType[4], MBR
alignedMBR)
{
	long i = 0, j = 0;
	long blockArea = sizeOfBlock * sizeOfBlock;

	for( i = alignedMBR.min.x; i < alignedMBR.max.x; i +=
sizeOfBlock )
	{
		for( j = alignedMBR.min.y; j < alignedMBR.max.y; j +=
sizeOfBlock )
		{
			Signature4CRS::Weight weight = block(i, j,
sizeOfBlock);
			nCellType[weight]++;
		}
	}

	return ( ( nCellType[Signature4CRS::Full] +
				nCellType[Signature4CRS::Strong] *
FACTOR_AREA_STRONG +
```

```
                        nCellType[Signature4CRS::Weak]              *
FACTOR_AREA_WEAK  ) * blockArea );
}

double                Signature4CRS::approximateOverlappingArea(const
Signature4CRS* signat4CRS2)
{
      double nOccupation[4][4] =
                  { { 0, 0, 0, 0 },
                    { 0, 0, 0, 0 },
                    { 0, 0, 0, 0 },
                    { 0, 0, 0, 0 } };
      return approximateOverlappingArea(signat4CRS2, nOccupation );
}

double                Signature4CRS::approximateOverlappingArea(const
Signature4CRS* signat4CRS2, double &deltaConfidence)
{
      long blockArea = sizeOfBlock * sizeOfBlock;
      double nOccupation[4][4] =
                  { { 0, 0, 0, 0 },
                    { 0, 0, 0, 0 },
                    { 0, 0, 0, 0 },
                    { 0, 0, 0, 0 } };

      double    area    =    approximateOverlappingArea(signat4CRS2,
nOccupation);

      double variance=0;
      deltaConfidence=0;
      for (int i=0; i<4; i++){
            for (int j=0; j<4; j++){
                  if(nOccupation[i][j] == 0)
                        continue;
                  if((i==1) && (j==1))
                        variance = VARIANCE_PxP;
                  else if(((i==1) && (j==2)) || ((i==2) && (j==1)))
                        variance = VARIANCE_PxM;
                  else if(((i==1) && (j==3)) || ((i==3) && (j==1)))
                        variance = VARIANCE_PxC;
                  else if((i==2) && (j==2))
                        variance = VARIANCE_MxM;
                  else
                        continue;
                  deltaConfidence     +=     (nOccupation[i][j]     *
CONFIDENCE_INTERVAL * sqrt(variance / nOccupation[i][j]));
            }
      }

      deltaConfidence *= blockArea;

      double areaMin = area - deltaConfidence;
      double areaMax = area + deltaConfidence;

      return area;
}

double                Signature4CRS::approximateOverlappingArea(const
Signature4CRS* signat4CRS2, double nOccupation[4][4] )
{
      if ( existsIntersection(this->map->mbr, signat4CRS2->map->mbr)
)
```

```
	{
		double blockArea;

		Coordinate min( MAX( this->map->mbr.min.x, signat4CRS2->map->mbr.min.x ), MAX( this->map->mbr.min.y, signat4CRS2->map->mbr.min.y ) );
		Coordinate max( MIN( this->map->mbr.max.x, signat4CRS2->map->mbr.max.x ), MIN( this->map->mbr.max.y, signat4CRS2->map->mbr.max.y ) );

		if (this->IsSmallerThan(signat4CRS2))
		{
			MBR interMBR(min, max);
			MBR alignedMBR = BBox2D::alignedMBR(interMBR, signat4CRS2->sizeOfBlock);

			return approximateOverlappingArea(this, signat4CRS2, alignedMBR, blockArea, nOccupation);
		}
		else
		{
			MBR interMBR(min, max);
			MBR alignedMBR = BBox2D::alignedMBR(interMBR, signat4CRS2->sizeOfBlock);

			return approximateOverlappingArea(signat4CRS2, this, alignedMBR, blockArea, nOccupation);
		}
	}
	return 0;
}


double Signature4CRS::approximateOverlappingArea(const Signature4CRS* smallerSignature, const Signature4CRS* largestSignature,

	MBR interMBR, double &blockArea, double nOccupation[4][4])
{
	// Matrix: Empty, Weak, Strong and Full
	double occupation[4][4] =
	{ { 0, 0, 0, 0 },
	  { 0, 0.0406, 0.1492, 0.1887 },
	{ 0, 0.1492, 0.6578, 0.8063 },
	{ 0, 0.1887, 0.8063, 1 } };

	double approximateArea = 0;

	for( long i = interMBR.min.x; i < interMBR.max.x; i += largestSignature->sizeOfBlock)
	{
		for( long j = interMBR.min.y; j < interMBR.max.y; j += largestSignature->sizeOfBlock)
		{
			Signature4CRS::Weight quadraA = smallerSignature->block(i, j, largestSignature->sizeOfBlock);
			Signature4CRS::Weight quadraB = largestSignature->block(i, j, largestSignature->sizeOfBlock);
			approximateArea += occupation[quadraA][quadraB];
			nOccupation[quadraA][quadraB] = nOccupation[quadraA][quadraB] + 1;
		}
```

```
	}

	blockArea = largestSignature->sizeOfBlock * largestSignature->sizeOfBlock;

	return approximateArea * blockArea;
}

double Signature4CRS::similar(Signature4CRS* signat4CRS2, double &resultMin, double &resultMax)
{
	double deltaConfidenceI = 0;
	double intersectionArea = this->approximateOverlappingArea(signat4CRS2, deltaConfidenceI);

	if(!intersectionArea)
		return 0;


	Signature4CRS* signat4CRSPlus = this->plus(signat4CRS2);

	double deltaConfidenceU = 0;
	double unionArea = signat4CRSPlus->approximateArea(deltaConfidenceU);

	if (unionArea == 0)
		return 0;

	resultMin = (intersectionArea - deltaConfidenceI) / (unionArea + deltaConfidenceU);
	resultMax = (intersectionArea + deltaConfidenceI) / (unionArea - deltaConfidenceU);

	double similarResult = intersectionArea / unionArea;

	return similarResult;
}

double Signature4CRS::similar(Signature4CRS* signat4CRS2)
{
	double intersectionArea = this->approximateOverlappingArea(signat4CRS2);

	if(!intersectionArea)
		return 0;

	double unionArea = this->plus(signat4CRS2)->approximateArea();

	return intersectionArea / unionArea;
}

Signature4CRS* Signature4CRS::plus(Signature4CRS* signat4CRS2)
{
	if ( existsIntersection(this->map->mbr, signat4CRS2->map->mbr) )
	{
		// Compute max and min coordinates of the union MBR
		Coordinate min( MIN( this->map->mbr.min.x, signat4CRS2->map->mbr.min.x ), MIN( this->map->mbr.min.y, signat4CRS2->map->mbr.min.y ) );
```

```
            Coordinate max( MAX( this->map->mbr.max.x, signat4CRS2-
>map->mbr.max.x  ),  MAX(  this->map->mbr.max.y,  signat4CRS2->map-
>mbr.max.y ) );

            if (this->IsSmallerThan(signat4CRS2))
            {
                  MBR unionMBR(min, max);
                  MBR   alignedMBR   =   BBox2D::alignedMBR(unionMBR,
signat4CRS2->sizeOfBlock);

                  return    plus(this,    signat4CRS2,    unionMBR,
alignedMBR);
            }
            else
            {
                  MBR unionMBR(min, max);
                  MBR   alignedMBR   =   BBox2D::alignedMBR(unionMBR,
this->sizeOfBlock);

                  return    plus(signat4CRS2,    this,    unionMBR,
alignedMBR);
            }
      }

      return this;
}

Signature4CRS*     Signature4CRS::plus(     const     Signature4CRS*
smallerSignature,
                                                      const
Signature4CRS* largestSignature, MBR unionMBR, MBR alignedMBR )
{
      long dx, dy;

      dx  =  (alignedMBR.max.x  -  alignedMBR.min.x)/largestSignature-
>sizeOfBlock;
      dy  =  (alignedMBR.max.y  -  alignedMBR.min.y)/largestSignature-
>sizeOfBlock;

      long numCells = dx * dy;

      if (numCells > PRODUCT) // PRODUCT is the maximum cells that
signature can have
            return NULL;

      RasterMap4CRS    rasterMap4CRS(1,    unionMBR,    dx,    dy,
largestSignature->map->potency);

      rasterMap4CRS.setGroupOfBits(Signature4CRS::Empty);

      MBR                 alignedMBRlargestSignat                 =
BBox2D::alignedMBR(largestSignature->map->mbr,     largestSignature-
>sizeOfBlock);

      //This  variables  are  used  to  compute  the  cell's  (x,  y)
coordinates of union signature considering the resolution change of
largest signature.
      long correcti, correctj;

      // This variables are used to compute the displacement between
the cell's position in the original sinature and the union signature
      long shiftX, shiftY;
```

```
	shiftX = (alignedMBRlargestSignat.min.x - alignedMBR.min.x);
	shiftY = (alignedMBRlargestSignat.min.y - alignedMBR.min.y);

	Signature4CRS::Weight weightL;

	for( long i = alignedMBRlargestSignat.min.x; i < 
alignedMBRlargestSignat.max.x; i += largestSignature->sizeOfBlock)
	{
		for( long j = alignedMBRlargestSignat.min.y; j < 
alignedMBRlargestSignat.max.y; j += largestSignature->sizeOfBlock)
		{
			correcti = ( (i - alignedMBRlargestSignat.min.x) + 
shiftX ) / largestSignature->sizeOfBlock;
			correctj = ( (j - alignedMBRlargestSignat.min.y) + 
shiftY ) / largestSignature->sizeOfBlock;
			weightL = largestSignature->block(i, j, 
largestSignature->sizeOfBlock);
			rasterMap4CRS.block(correcti, correctj, weightL);
		}
	}

	MBR alignedMBRsmallerSignat = 
BBox2D::alignedMBR(smallerSignature->map->mbr, largestSignature-
>sizeOfBlock);

	shiftX = (alignedMBRsmallerSignat.min.x - alignedMBR.min.x);
	shiftY = (alignedMBRsmallerSignat.min.y - alignedMBR.min.y);

	Signature4CRS::Weight weightU, weightS;

	for( long i = alignedMBRsmallerSignat.min.x; i < 
alignedMBRsmallerSignat.max.x; i += largestSignature->sizeOfBlock)
	{
		for( long j = alignedMBRsmallerSignat.min.y; j < 
alignedMBRsmallerSignat.max.y; j += largestSignature->sizeOfBlock)
		{
			correcti = ( (i - alignedMBRsmallerSignat.min.x) + 
shiftX ) / largestSignature->sizeOfBlock;
			correctj = ( (j - alignedMBRsmallerSignat.min.y) + 
shiftY ) / largestSignature->sizeOfBlock;
			weightU = rasterMap4CRS.block(correcti, correctj);
			weightS = smallerSignature->block(i, j, 
largestSignature->sizeOfBlock);
			if ( (weightU == Signature4CRS::Empty) || 
(weightS == Signature4CRS::Full) )
				rasterMap4CRS.block(correcti, correctj, 
weightS);
			else if ( weightU == Signature4CRS::Weak && 
weightS == Signature4CRS::Strong )
				rasterMap4CRS.block(correcti, correctj, 
weightS);
		}
	}

	Signature4CRS* unionSignature4CRS = new 
Signature4CRS(rasterMap4CRS);

	return unionSignature4CRS;
}
```

```
Signature4CRS*            Signature4CRS::intersection(Signature4CRS*
signat4CRS2)
{
      if ( existsIntersection(this->map->mbr, signat4CRS2->map->mbr)
)
      {
            Coordinate min( MAX( this->map->mbr.min.x, signat4CRS2-
>map->mbr.min.x  ),  MAX(  this->map->mbr.min.y,  signat4CRS2->map-
>mbr.min.y ) );
            Coordinate max( MIN( this->map->mbr.max.x, signat4CRS2-
>map->mbr.max.x  ),  MIN(  this->map->mbr.max.y,  signat4CRS2->map-
>mbr.max.y ) );

            if  (this->IsSmallerThan(signat4CRS2))  //signat4CRS2  is
greater than this
            {
                  MBR interMBR(min, max);
                  MBR  alignedMBR  =  BBox2D::alignedMBR(interMBR,
signat4CRS2->sizeOfBlock);

                  return  intersection(this,  signat4CRS2,  interMBR,
alignedMBR);
            }
            else //signat4CRS2 is smaller than this
            {
                  MBR interMBR(min, max);
                  MBR  alignedMBR  =  BBox2D::alignedMBR(interMBR,
this->sizeOfBlock);

                  return  intersection(signat4CRS2,  this,  interMBR,
alignedMBR);
            }
      }

      return NULL;
}

Signature4CRS*  Signature4CRS::intersection(  const  Signature4CRS*
smallerSignature,

      const  Signature4CRS*  largestSignature,  MBR  interMBR,  MBR
alignedMBR )
{
      long dx, dy;

      dx  =  (alignedMBR.max.x  -  alignedMBR.min.x)/largestSignature-
>sizeOfBlock;
      dy  =  (alignedMBR.max.y  -  alignedMBR.min.y)/largestSignature-
>sizeOfBlock;

      RasterMap4CRS     rasterMap4CRS(1,     interMBR,     dx,     dy,
largestSignature->map->potency);

      rasterMap4CRS.setGroupOfBits(Signature4CRS::Empty);

      MBR                  alignedMBRlargestSignat                  =
BBox2D::alignedMBR(largestSignature->map->mbr,     largestSignature-
>sizeOfBlock);
      MBR                  alignedMBRsmallerSignat                  =
BBox2D::alignedMBR(smallerSignature->map->mbr,     largestSignature-
>sizeOfBlock);
```

```
	long correcti, correctj;
	for( long i = alignedMBR.min.x; i < alignedMBR.max.x; i += largestSignature->sizeOfBlock)
	{
		for( long j = alignedMBR.min.y; j < alignedMBR.max.y; j += largestSignature->sizeOfBlock)
		{
			correcti	=	(i	-	alignedMBR.min.x)	/ largestSignature->sizeOfBlock;
			correctj	=	(j	-	alignedMBR.min.y)	/ largestSignature->sizeOfBlock;
			Signature4CRS::Weight weightS = smallerSignature->block(i, j, largestSignature->sizeOfBlock);
			Signature4CRS::Weight weightL = largestSignature->block(i, j, largestSignature->sizeOfBlock);

			if (weightL == Signature4CRS::Full)
				rasterMap4CRS.block(correcti,	correctj, weightS);
			else if (weightS == Signature4CRS::Full)
				rasterMap4CRS.block(correcti,	correctj, weightL);
			else if ( (weightL == Signature4CRS::Empty) || (weightS == Signature4CRS::Empty) )
				rasterMap4CRS.block(correcti,	correctj, Signature4CRS::Empty);
			else if ( (weightL == Signature4CRS::Strong) && (weightS == Signature4CRS::Strong) )
				rasterMap4CRS.block(correcti,	correctj, Signature4CRS::Strong);
			else /* ( (b.type == Strong) and (s.type==Weak) ) or ( (s.type == Strong) and (b.type==Weak) ) or ( (b.type == Weak) and (s.type==Weak) ) */
				rasterMap4CRS.block(correcti,	correctj, Signature4CRS::Weak);
		}
	}

	Signature4CRS*		unionSignature4CRS		=		new Signature4CRS(rasterMap4CRS);

	return unionSignature4CRS;
}

bool Signature4CRS::IsSmallerThan(const Signature4CRS* signat4CRS2)
{
	return (this->map->potency < signat4CRS2->map->potency);
}
```