Universidade Estadual de Londrina

OLAVO PEREIRA DO NASCIMENTO

TÉCNICAS DE ARMAZENAMENTO E CONSULTA DE DADOS DE PRECIPITAÇÃO OBTIDOS POR MEIO DE IMAGENS DE RADAR

LONDRINA

2023

OLAVO PEREIRA DO NASCIMENTO

TÉCNICAS DE ARMAZENAMENTO E CONSULTA DE DADOS DE PRECIPITAÇÃO OBTIDOS POR MEIO DE IMAGENS DE RADAR

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação do Departamento de Computação da Universidade Estadual de Londrina, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Supervisor: Prof. Dr. Daniel dos Santos Kaster

LONDRINA

2023

OLAVO PEREIRA DO NASCIMENTO

TÉCNICAS DE ARMAZENAMENTO E CONSULTA DE DADOS DE PRECIPITAÇÃO OBTIDOS POR MEIO DE IMAGENS DE RADAR

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação do Departamento de Computação da Universidade Estadual de Londrina, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

Orientador: Prof. Dr. Daniel dos Santos Kaster
Universidade Estadual de Londrina

Prof. Dr. Adilson Luiz Bonifácio
Universidade Estadual de Londrina

Dr. Pablo Ricardo Nitsche
IDR-Paraná

Londrina, 15 de maio de 2023.

ACKNOWLEDGEMENTS

Agradeço meu orientador e professor Daniel dos Santos Kaster, pelo auxílio, paciência e sabedoria durante o desenvolvimento deste trabalho. Agradeço o corpo docente do Departamento de Computação, por proporcionarem um ambiente propício para o desenvolvimento de novos profissionais capacitados para atender as necessidades da sociedade. Agradeço a minha família e colegas por me acompanharem durante esse longo trajeto.

"O homem não teria alcançado o possível se, repetidas vezes, não tivesse tentado o impossível.
(Max Weber)

NASCIMENTO, OLAVO. **Técnicas De Armazenamento e Consulta De Dados De Precipitação Obtidos Por Meio De Imagens De Radar**. 2023. 46f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2023.

ABSTRACT

Com o crescente uso de dados climáticos em múltiplas ferramentas meteorológicas, torna-se evidente a necessidade de conjuntos de dados eficientes e de fácil utilização. Contudo, o estado do Paraná não possui dados de precipitação para uso acadêmico e comercial que sejam facilmente integrados a modelos preditivos. Este trabalho tem como objetivo utilizar dados de precipitação capturadas através de radares do SIMEPAR, os quais são posteriormente processados através da segmentação das imagens e então armazenadas de forma a minimizar o espaço utilizado, sem renunciar a eficiência de busca. Visando a criação de um banco de dados que possa auxiliar outras áreas que carecem de dados apropriados, em especial a agronômica. O conjunto de dados resultante é, ao mesmo tempo, pequeno e fácil de consultar, permitindo a recuperação rápida de dados num ponto específico no espaço e no tempo.

Palavras-chave: Segmentação de Imagens. Armazenamento de Dados de Precipitação. Modelagem Espaço-Temporal. Consultas Espaço-Temporais. GIS

NASCIMENTO, OLAVO. **Techniques for Storing and Querying Precipitation Data from Radar Imagery**. 2023. 46p. Final Project (Bachelor of Science in Computer Science) – State University of Londrina, Londrina, 2023.

ABSTRACT

With the increasing use of climate data in multiple meteorological tools, the need for efficient and user-friendly datasets becomes evident. However, the state of Paraná lacks precipitation data that can be easily integrated into predictive models for academic and commercial use. This work aims to use precipitation data captured through SIMEPAR radars, which are further processed through image segmentation and then stored in a way that minimizes the space used, without renouncing search efficiency. Thus aiming to create a database that can help other areas that lack appropriate data, especially agronomy. The resulting dataset is at the same time small and easy to query, allowing for quick retrieval of data in a specific point in space and time.

Keywords: Image Segmentation. Precipitation Data Storage. Spatial-Temporal Data Modelling. Spatial-Temporal Queries. GIS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

EQP	Estimativa quantitativa de precipitação
JPEG	Joint Photographic Experts Group
SIMEPAR	Sistema Meteorológico do Paraná
ED	Evolução Diferencial
SIRGAS	Sistema de Referencia Geocêntrico para as Américas
GDAL	Geospatial Data Abstraction Library
GIS	Geographic Information System
SRID	Spatial Reference ID
CUAHSI	Consortium of Universities for the Advancement of Hydrologic Science, Inc.

CONTENTS

1 INTRODUÇÃO

Uma das principais entidades de coleta, análise e distribuição de dados climáticos no estado do Paraná é o SIMEPAR (Sistema de Tecnologia e Monitoramento Ambiental do Paraná), o qual utiliza três radares meteorológicos, posicionados em Teixeira Soares, Cascavel e Curitiba, assim como o uso de satélites para para capturar dados atmosféricos.[1]

A ideia de utilizar radares para mapear dados de precipitação em mapas é antiga, tendo origem na década de 1940 [2]. Na atualidade, uma época pós corrida espacial, onde ferramentas baseadas no uso satélites se tornaram algo comum no cotidiano de todos, dados de radares continuam relevantes. Um dos motivos para essa constatação é intrínseco à forma com que atuam. Enquanto satélites medem condições através da atmosfera, radares meteorológicos são capazes de medir as condições na superfície do planeta, permitindo extrair informações com maior resolução espaço-temporal [3].

Os radares utilizados pelo SIMEPAR operam em frequências entre 2,7GHz a 2,9GHz, sendo categorizadas como bandas de tipo S, o que pode ser verificado na tabela 1 (adaptada de [4]). A área de captura representa uma área de 480km ao redor dos radares, permitindo capturar não só todo o Paraná, como Santa Catarina, o sul do estado de São Paulo e o norte do Rio Grande do Sul [5].

Table 1 – Frequência de Banda de Radares

Banda	Frequência (GHz)	Tamanho da Onda (cm)
Milímetro	40-100	0.75-0.30
Ka	26.5-40	1.1-0.75
K	18-26.5	1.7-11
Ku	12.5-18	2.4-17
X	8-12.5	3.75-2.4
C	4-8	7.5-3.75
S	2-4	15-7.5
L	1-2	30-15
UHF	0.3-1	100-30

Como evidenciado em [6], dados de precipitação com boa resolução espaço-temporal são necessários em escala global. Todavia, ainda que os radares capturem dados em um curto intervalo de tempo, os dados gerados não são de fácil utilização por terceiros. Entre os motivos que dificultam este processo, dois são de notável importância. Primariamente, a abundância de dados atmosféricos capturados diariamente pelos radares torna inviável a classificação manual [17], já que os dados gerados excedem a velocidade de processamento disponível a um indivíduo ou grupo de pesquisa. Ademais, mesmo com a grande

[1] <http://www.simepar.br>

quantidade disponíveis para estudos, grande parte se torna proibitiva do ponto de vista da mineração de dados por utilizarem formatos de distribuição e consulta que inviabilizam consultas sobre os dados capturados [8][9][10].

Mais especificamente para este trabalho, o acesso a informações já capturadas por entidades públicas e privadas se provou o principal empecilho para a reutilização de bancos de dados existentes. Já que os dados existentes não podem ser acessados de forma simples, necessitando a requisição de um período de captura de dados para as entidades.

Buscando mitigar esse problema em uma escala local, o objetivo deste trabalho é construir um banco de dados de precipitação a partir de dados de radar meteorológico com escopo limitado ao estado do Paraná. O processo de construção deste banco de dados inclui o processamento das imagens capturadas, a estruturação dos dados trabalhados de forma compacta e adequada, assim como a utilização de tecnologias já estabelecidas como PostGIS para a realização de consultas espaço-temporais, visando disponibilizar dados de qualidade apropriada para uso no treinamento de modelos preditivos empregados na área da agronomia.

Foi definido como caso de estudo as imagens disponibilizadas pelo SIMEPAR, contudo o processo pode ser em grande parte reaproveitado e aplicado a outras fontes de dados que também forneçam imagens de radares meteorológicos. Após o processamento e inserção dos dados coletados, o conjunto de dados resultantes se mostrou compacto, do ponto de vista de armazenamento. Ao mesmo tempo que permite toda a expressividade e simplicidade da linguagem SQL para a realização de consultas, utilizando diversas otimizações já inseridas no planejador de consultas do PostgreSQL.

Este trabalho é estruturado da seguinte forma. A seção 2 apresenta os fundamentos necessário para o entendimento deste trabalho e de trabalhos correlatos. A seção 3 apresenta a proposta de solução do projeto. A seção 4 descreve os resultados obtidos. Por fim, a seção 5 apresenta a conclusão e possíveis melhorias a serem realizadas em trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICO-METODOLÓGICA

2.1 Segmentação de Imagens

Segmentação de imagens é um dos principais problemas abordados nas áreas de visão computacional e processamento de imagens. Podendo ser definido como o processo de identificação e delineação de objetos em imagens, sendo uma etapa crucial para operações realizadas sobre imagens de interesse [11][12]. A importância desse processo na análise de dados se torna ainda mais evidente ao considerarmos que todos os dados finais, assim como os resultados e conclusões encontradas ao analisa-los, dependem do processamento realizado sobre as imagens iniciais [11][12][13].

O processo de segmentação pode ser separado em duas partes: reconhecimento e delineação. A primeira se baseia em encontrar o objeto de interesse na imagem fornecida, sendo um processo de alto nível. Após encontrar o objeto de interesse é preciso separa-lo das partes da imagem que não possuem pontos de interesse, para isso a delineação, um processo de baixo nível, determina o tamanho de cada um dos pontos de interesse encontrados anteriormente [11].

Como visto em [11][13], algoritmos gerais para segmentação não são viáveis para a grande maioria dos casos de estudos que precisam utilizar este tipo de técnica. Isso acontece devido a necessidade de conhecimento específico ao dados abordados, permitindo que algoritmos feitos com base no conhecimento humano dos dados e guiado em experimentos que buscam aproximar a segmentação automatizada da segmentação manual feita por um especialista.

2.1.1 Segmentação Baseada em Thresholding Multi-Nível

Thresholding é uma das técnicas de segmentação de imagens que se provou extremamente popular e com diversos modos de aplicação, ao mesmo tempo que possui um conceito extremamente simples ao delimitar um valor limite para definir se um ponto da imagem deve ser ativado ou desativado [14]. Para definir esse valor limite pode-se utilizar como métrica valores globais, como o histograma da imagem analisada ou valores locais, como uma matriz de coocorrência.

Técnicas de segmentação baseadas em *thresholding* podem ser divididas em duas categorias: *thresholding* de dois níveis e *thresholding* multi-nível [13][14]. No primeiro caso a imagem é separada em duas regiões, normalmente o objeto desejado em preto e o fundo em branco. Isso limita o algoritmo a somente essas classificações, o que pode afetar os resultados de para imagens com mais de um tipo de classe de objetos.

Já a segmentação multi-nível atua através de múltiplos *thresholds* aplicados a subdivisões da imagem, buscando diferenciar duas ou mais classes de objetos na imagem através de uma métrica que pode variar entre diferentes casos [12][13]. Casos onde a imagem possui ruído ou a iluminação é ruim são exemplos que podem se beneficiar dessa abordagem.

Dessa forma, um algoritmo de segmentação pode ser visto como um problema de classificação de objetos de interesse para classes pré-definidas, sendo preciso definir a abordagem mais apropriada de acordo com a imagem que se deseja segmentar, tendo em vista que não é possível generalizar algoritmos de segmentação para todos os casos [11][13] e nem estabelecer a segmentação perfeita para uma imagem de entrada [11].

2.2 Evolução Diferencial

Evolução Diferencial é um algoritmo de otimização estocástico baseado em população [15]. Os parâmetros que devem ser otimizados são representados como um vetor, o qual quando aplicado em uma função de custo objetiva e modelada de acordo com os resultados que se deseja obter, retorna uma pontuação individual que deve ser minimizada, permitindo a comparação objetiva entre os diferentes vetores com base nas métricas definidas [16]. A eficiência do algoritmo é dependente dos parâmetros de controle utilizados e na estratégia de geração de novos vetores através de mutações [17].

A principal diferença da evolução diferencial de outros algoritmos evolutivos, como algoritmos genéticos, é o foco da evolução diferencial na função de mutação utilizada para aumentar a diversidade da população, já no caso de algoritmos genéticos o foco principal é a função de *crossover*. Também é a vantagem na velocidade com que algoritmos de evolução diferencial convergem e o menor número de parâmetros que devem ser controlados quando comparados com algoritmos genéticos [14].

Assim como outros algoritmos evolutivos, a evolução diferencial possui quatro principais estágios: inicialização, calculo de *fitness*, reprodução e fim [12]. A definição inicial da população é feita de forma aleatória $X_{i,G}(i = \{1, 2, ..., N_p\})$, onde G representa a geração da população, N_p o número de indivíduos na população e cada indivíduo possui N_d vetores [18][16].

Para que novos valores possam ser incorporados a população inicial é necessário que a mesma sofra um processo de mutação. Para as versões clássicas da evolução diferencial esse processo envolve a criação de um novo indivíduo $V_{i,G}$ com base em outros 3 vetores da população.

$$V_{i,G} = X_{r1,G} + F(X_{r2,G} - X_{r3,G}) \tag{2.1}$$

Onde $r1$, $r2$ e $r3$ são 3 índices diferentes entre si e diferentes de i escolhidos aleatoriamente da população original e F é uma constante de amplificação. Maiores valores

de F aumentam a diversidade da população pós-mutação, buscando evitar que o algoritmo fique preso em resultados de mínimas locais. Valores menores resultam em uma convergência mais rápida do resultado [18].

Após o processo de mutação, é realizado o *crossover* entre a população original e a população gerada no passo anterior. A ideia é aumentar a variedade dos indivíduos da população trocando alguns vetores entre os cromossomos de cada população. A definição de quais vetores são alterados é realizada de forma aleatória.

$$U_{ji,G} = \begin{cases} V_{ji,G} & rand_j(0,1) < C_r \\ X_{ji,G} & seno \end{cases} \tag{2.2}$$

Onde $C_r \in [0,1]$ é uma constante representando a probabilidade de um elemento da nova população sofrer crossover.

Por fim, são selecionados os indivíduos com a menor pontuação calculada através da função objetiva f definida para o caso abordado.

$$U_{ji,G+1} = \begin{cases} V_{ji,G+1} & f(V_{i,G+1}) <= f(X_{i,G}) \\ X_{ji,G} & seno \end{cases} \tag{2.3}$$

Esse processo é repetido até que a condição de parada seja atendida. Muitas vezes a condição para o fim do algoritmo é um numero de iterações pré-definido.

2.3 PostGIS

PostGIS é uma extensão *open-source* para o banco de dados relacional PostgreSQL que adiciona suporte a objetos geográficos, assim como funções para manipular e consultar esses dados[19]. Por utilizar uma base sólida como o PostgreSQL, vários pontos já desenvolvidos para uso geral podem ser reaproveitados e estendidos, exemplos são o suporte a transações, índices que podem ser aplicados a objetos de tipo espacial e o planejador de consultas. Sendo notável a capacidade de extensão do banco PostgresSQL através de programas de terceiros [20]. Utilizando as diversas funções inclusas é possível medir distâncias e áreas, determinar relacionamentos espaciais entre objetos e realizar junções espaciais entre tabelas com dados espaciais.

2.4 Trabalhos Correlatos

2.4.1 Segmentação de Imagens

Existem diversos artigos utilizando algoritmos evolutivos para definir os valores de segmentação. Em [12] algoritmos genéticos são utilizados para abordar um problema de segmentação de imagens como um problema de otimização de parâmetros de threshold.

Isso se mostrou necessário devido a complexidade para chegar ao resultado final. Se t thresholds precisem ser encontrados a complexidade de um algoritmo de força bruta se torna muito grande, sendo possíveis $L \times (L-1) \times ... \times (L-t+1)$ combinações. Onde L é o valor máximo de cor, para imagens de cinza esse valor seria 256.

No que tange algoritmos de evolução diferencial, diversas variações do algoritmo original foram discutidas, apresentando as vantagens e desvantagens de cada uma. Já em [15], o autor apresenta uma implementação do algoritmo de evolução diferencial DE/rand/1/bin. Também são apresentados testes subjetivos e objetivos realizados sobre os resultados de segmentação para comparar o algoritmo implementado com o algoritmo de Kittler.

Para a implementação de evolução diferencial apresentada em [21] a entropia difusa da imagem é definida como a função de otimização. Também são disponibilizadas diferentes formas de calcular a entropia, assim como as diferenças entre os resultados das diferentes opções.

Com foco específico a área da saúde [22] apresenta o uso de evolução diferencial para realizar a segmentação de imagens de ferimentos na pele. O algoritmo foi escolhido devido sua tendência a convergir rapidamente, baixo número de parâmetros a serem definidos e ajustados, assim como a facilidade de alcançar um ponto de ótima global. Contudo, foi necessário aplicar etapas de pré-processamento nas imagens para auxiliar a o processo de segmentação.

O autor em [14] aborda a utilização de algoritmos de evolução diferencial na segmentação multi-nível de imagens através da análise de histogramas de escala de cinza na imagem. A adição da evolução diferencial permite que o algoritmo evite ficar preso em pontos de mínimas locais através da mutação de indivíduos, sem sacrificar o desempenho de execução.

Por fim, o artigo mais relevante para o tópico abordado nesse trabalho foi apresentado em [17]. Onde o autor discute a segmentação de imagens de precipitação capturadas através de radares meteorológicos, podendo ser realizada a segmentação com base em diferentes tipos de eventos de precipitação. Para isso é proposto uma variação da função de mutação para os indivíduos da população, enquanto que as demais funções do algoritmo permanecem iguais a implementação mais comum. A nova função de mutação pode ser definida como:

$$V_{i,G} = X_{r1,G} + F(N1(X_{g,best} - X_{r2,G}) - N2(X_{g,best} - X_{r3,G})) \tag{2.4}$$

Onde r1, r2 e r3 continuam sendo índice aleatórios da população e $X_{g,best}$ representa o indivíduo com melhor *fitness* da população. N_1 é um valor aleatório on de $N_1 \in [0, 1]$ e $N2$ é definido como o complemento de N_1. Ao considerar o melhor indivíduo da população é possível acelerar o número de iterações necessárias para que ocorra a convergência do

resultado e permite que a população criada a partir do processo de mutação não se limite apenas a valores ótimos locais.

2.4.2 Armazenamento e Consulta de Dados Espaço Temporais

Em [10] os autores abordam o processo de transformação e visualização de dados de precipitação que utiliza diversos formatos de arquivos durante o fluxo de execução, entre os formatos utilizados estão: XMRG binário, uma grade ASCII e por fim um formato GIS em grade porém proprietário. O objetivo final é definir um fluxo de processamento de dados para os dados do conjunto de dados NEXRAD Stage III capaz de automaticamente transformar os dados de entrada em um formato GIS para utilização em pesquisas.

Já em [8], a abordagem utilizada se baseia em analisar dados de precipitação de forma manual, filtrando dados desnecessários e alcançando uma excepcional redução de armazenamento de 99% quando comparado aos dados originais. Para isso foram implementadas 5 tabelas especificadas no padrão de modelagem de dados observacionais CUAHSI, com foco na modelagem de dados no campo da hidrologia.

São definidos conceitos como tempestades locais, horárias e gerais para mapear de forma expressiva e eficiente os eventos de precipitação durante o processo de análise de tempestades. O algoritmo pode ser divido em quatro partes principais, a separação de eventos, analise de localização e proximidade, identificação de uma sub-tempestade e identificação de uma tempestade principal.

Em [23] o autor segue uma linha próxima ao proposto em [8], entretanto o modelo proposto tem como base as propostas de outros 3 trabalhos prévios, visando tornar o modelo proposto mais resiliente ao lidar com fenômenos dinâmicos como precipitação, enchentes e queimadas.

Para isso são definidos 3 tipos principais de objetos: estados, processos e eventos. Estados representam os pontos de precipitação em um período de tempo T. Processos é representado por um grupo de estados e registra a mudança no tempo dos estados. Enquanto isso, um evento é o objeto com maior resolução espaço-temporal por ser composto por vários processos e sendo utilizado para registrar o tempo inicial e final dos processos que o compõem.

Contudo, a abordagem mais próxima da adotada foi a exemplificada em [24], onde os autores utilizam imagens raster de forma direta através das funcionalidades do PostGIS, sem a necessidade de passos intermediários de análise de eventos climáticos. Por utilizar a biblioteca geoespacial GDAL é possível carregar os mais populares formatos de imagens raster georreferenciada de forma simples e direta.

A inserção das imagens raster é feita através da ferramenta de linha de comando raster2pgsql, a qual lê uma imagem georreferenciada e gera o código SQL equivalente para

realizar sua inserção sem nenhuma perda de dados. Também é abordado a consulta dos dados que foram inseridos através das diversas funções disponibilizadas pelo PostGIS, permitindo buscar pontos de intersecção, distância entre pontos, lidar com diferentes sistemas geográficos, entre outras possibilidades.

3 PROPOSTA DE SOLUÇÃO: SEGMENTAÇÃO E ARMAZENAMENTO DE DADOS DE PRECIPITAÇÃO

3.1 Descrição do Conjunto de Dados Trabalhado

Os dados de precipitação abordados nesse trabalho são imagens de radares meteorológicos advindas do *website* do SIMEPAR e coletadas através de várias estações dispersas pelo estado, como visto na figura 1.

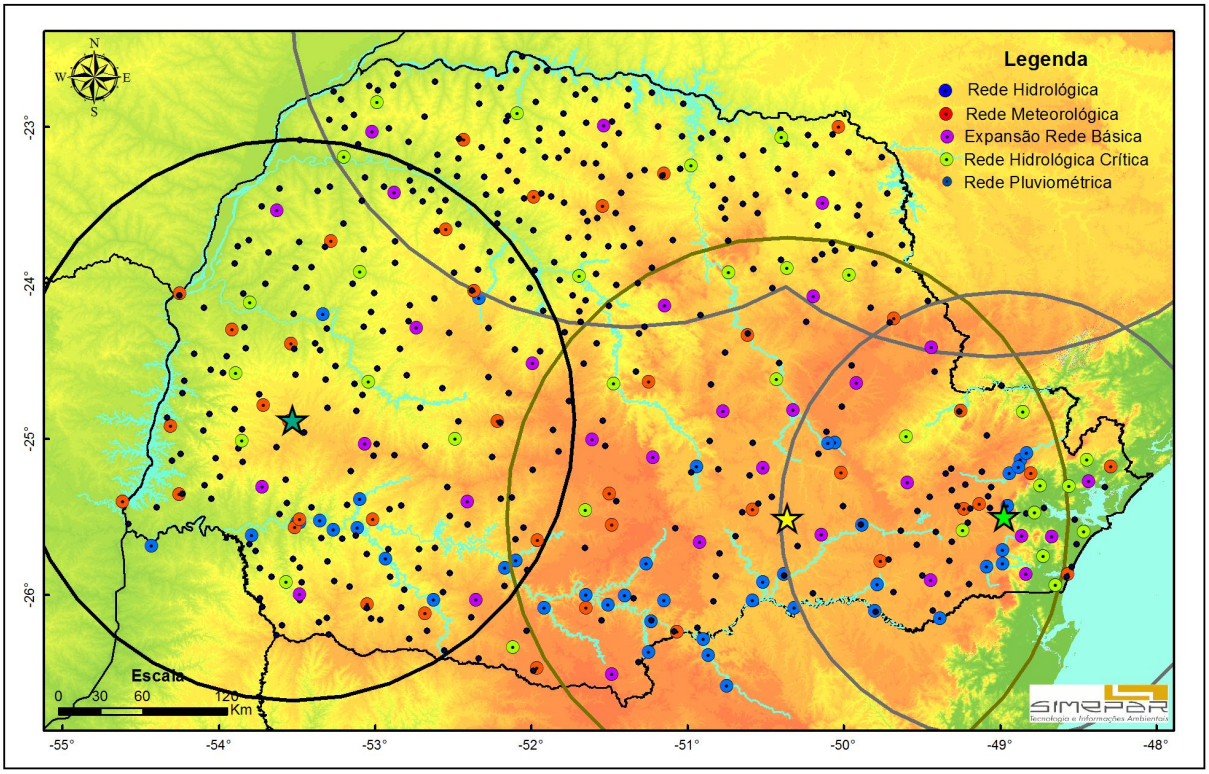


Figure 1 – Visão geral do sistema de monitoramento do SIMEPAR[1].

Cada imagem representa uma estimativa quantitativa de precipitação (EQP) para toda a área do estado, assim como alguns pontos fora do estado. O que permite um rápido entendimento visual de toda a precipitação acumulada em uma janela de tempo específica[25], para os dados estudados esta janela de tempo é de 10 minutos.

A fim de facilitar a leitura humana, os dados de precipitação são apresentados como imagens em formato JPEG, onde diferentes cores do gradiente indicam valores específicos de precipitação para cada pixel na imagem, assim como visto na figura 2. Porém, como apontado em [10] e [8] é comum que dados de precipitação sejam armazenados em formatos que não são facilmente analisados e minerados, ao mesmo tempo que também utilizam muito espaço em disco. Pontos que se provam empecilhos para a utilização dos dados para pesquisas em outras áreas e instituições.

A atualização dos dados disponíveis é realizada a cada 10 minutos, sendo rapidamente distribuída através da internet. Contudo, é limitada a um histórico das últimas 8 imagens coletadas, inviabilizando a análise de dados capturados em um período anterior a 80 minutos somente utilizando as informações disponíveis no momento.

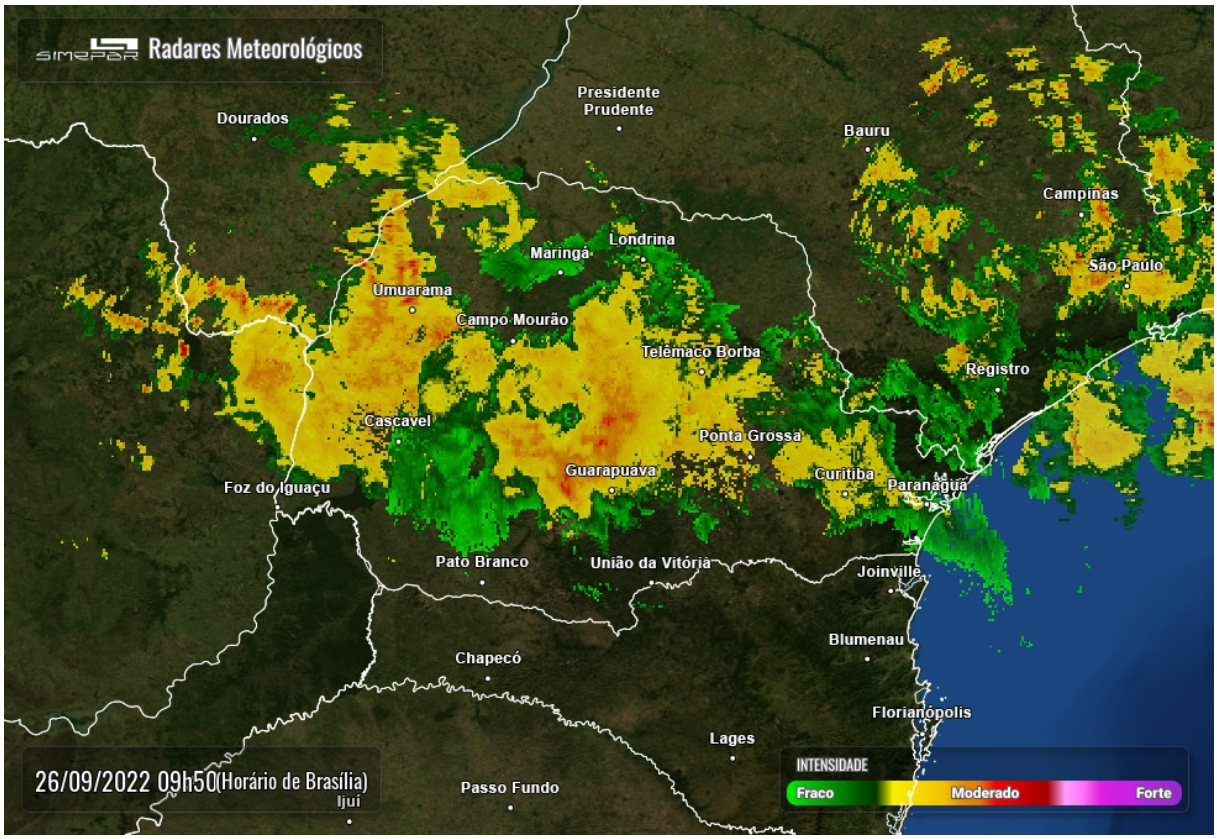


Figure 2 – Exemplo de imagem de radar disponibilizada pelo SIMEPAR.

3.2 Pré-processamento de Imagens

3.2.1 Segmentação

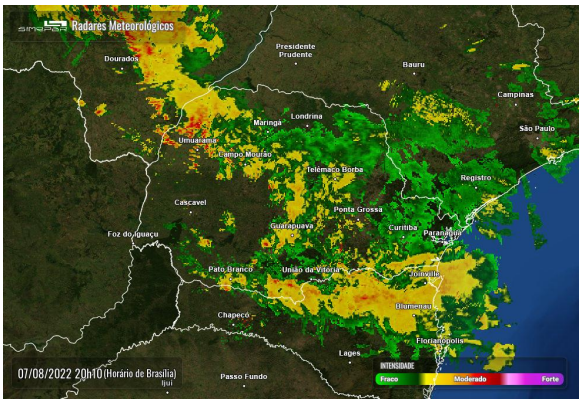
Como o plano de fundo da imagem pode algumas vezes coincidir com as cores do gradiente, é essencial categorizar a imagem em dois grupos: os pontos de precipitação e o fundo da imagem. Como em [22], foi realizada uma etapa anterior (Algoritmo 1) a segmentação para alcançar melhores resultados finais. Isso ocorre devido aos textos e informações adicionais presentes em cada imagem de radar que requerem atenção especial, uma vez que os valores de reflexividade nestes pontos ficam cobertos por linhas brancas inseridas para facilitar a visualização humana da imagem.

Como esses dados não são necessários para o conjunto de dados final, os mesmos foram removidos através do uso de uma máscara feita de forma manua e seus valores são preenchidos com base nas cores ao redor, buscando aproximar os valores originais

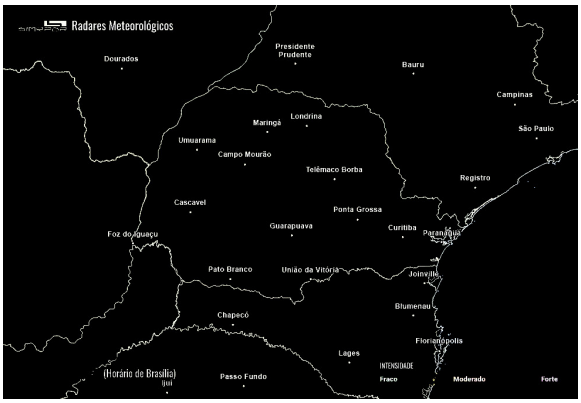
observado pelos radares. Essa abordagem que se mostrou efetiva, como pode ser visto na figura 3.

Algoritmo 1 Algoritmo Pré-Segmentação

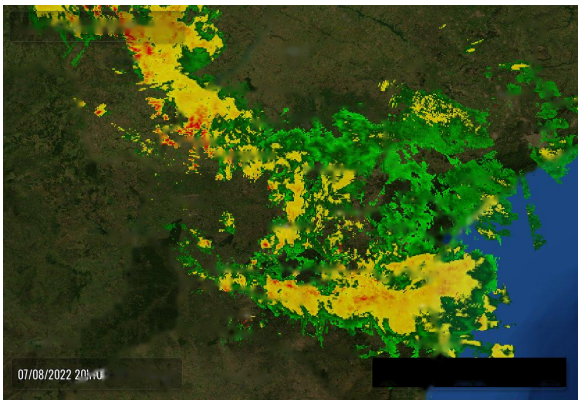
$imagem[600:650, 630:960] \leftarrow 0$ ▷ Remove o gradiente na parte inferior direita da imagem
$mascara \leftarrow abrir\_imagem('title\_mask.png')$ ▷ Máscara preta e branca contendo somente os títulos e divisas entre estados e países (Figura 6b)
$imagem \leftarrow inpaint(image, mascara)$ ▷ Utiliza a mascara especificada para remover os pontos demarcados e preenche-los com valores ao redor na imagem original (Figura 6d)



(a) Imagem de radar original



(b) Máscara



(c) Imagem de radar sem divisas estaduais, texto superior e gradiente

Figure 3 – Exemplo de pré-processamento de imagem

Após analisar as diferentes opções, a segmentação das imagens foi realizada utilizando o algoritmo descrito em [17]. Essa escolha foi feita levando em consideração a similaridade entre as imagens de radar a serem segmentadas nos dois trabalhos, onde ambas representam a reflexidade de partículas em uma área ao redor dos radares através de diferentes cores em um gradiente.

Ademais, assim como apontado pelos autores, o algoritmo demonstrou tempo computacional apropriado e resultados finais de qualidade satisfatória para realizar o proces-

samento das imagens capturadas. O desempenho do algoritmo é de imensa importância para determinar a opção mais apropriada, tendo em vista que após o período de coleta de 9 meses foram totalizadas 27247 imagens meteorológicas.

A evolução diferencial segue um processo similar a de outros tipo de algoritmos genéticos [15][22], onde uma população inicial randomizada é submetida a várias iterações onde 4 operações de operações básicas são realizas sobre indivíduos de uma população, são elas: mutação, crossover, seleção e calculo de *fitness* dos indivíduos. O algoritmo 2 apresenta o processo de evolução diferencial aplicado a segmentação de imagens através da otimização de thresholds.

Algoritmo 2 Algoritmo para Segmentação das Imagens Utilizando Evolução Diferencial

População inicializada com valores aleatórios. ▷ A população possui N_p indivíduos com N_d vetores (thresholds) cada
$fitness \leftarrow fitness(population)$ ▷ *Fitness* da população inicial é calculada (Equação 3.1)
$i \leftarrow 0$
while $i < n\_iterations$ **do**
  $mutated\_pop \leftarrow mutation(population)$ ▷ Equação 2.4
  $crossover\_pop \leftarrow crossover(population, mutated\_pop)$ ▷ Equação 2.2
  $population \leftarrow select(population, crossover\_pop)$ ▷ Equação 2.3
  $fitness \leftarrow fitness(population)$ ▷ Equação 3.1
  $i \leftarrow i + 1$
end while

A priori, para tornar o processo de segmentação um problema de otimização a ser resolvido pela evolução diferencial é necessário definir como o problema será representado através de um cromossomo [12][22]. Em [12], [17] e [18] os autores definem que cada vetor de um indivíduo representa um threshold referente a um pedaço da imagem original e que deve ser otimizado pelo algoritmo. Portanto uma imagem de entrada I será dividida em de N_d sub-divisões, onde cada indivíduo da população possui N_d thresholds, visando diferenciar o maior número de classes possíveis através de múltiplos thresholds [18].

A função de *fitness* adotada pelos autores em [17] busca minimizar a entropia presente na imagem, podendo ser definida através da seguinte função:

$$entropy = \frac{1}{IJln2} \sum_{i=1}^{I} \sum_{j=1}^{J} S_n(\mu_y(y_{ij})) \tag{3.1}$$

Onde $\mu_y(y_{ij})$ representa o brilho da imagem y na posição ij e S_n é o resultado da função de Shannon para calculo de informação no ponto, podendo ser definida como:

$$S_n(\mu_y(y_{ij})) = -\mu_y(y_{ij})ln(\mu_y(y_{ij})) - (1 - \mu_y(y_{ij})ln(\mu_y(y_i))) \tag{3.2}$$

As funções de crossover (equação 2.2) e seleção (equação 2.3) são idênticas as utilizadas em algoritmos genéticos genéricos. Enquanto que a função de mutação adotada (equação 2.4) inclui as alterações sugeridas em [17] para acelerar o tempo necessário para a convergência de um resultado.

3.2.2 Extração dos Valores Numéricos

As imagens de radar estudadas são feitas para facilitar a leitura humana dos dados dados numéricos capturados pelos radares, sem fornecer uma maneira de realizar o processo reverso, extrair dados numéricos das imagens, dificultando a utilização dessas informações por terceiros. Como proposto em [26] e especificado no website do SIMEPAR, as imagens de radar apresentam a reflexividade das partículas de precipitação na área de alcance através de um gradiente de cores.

Como os valores de todas as cores não são conhecidos foi necessário estimar valores para cores entre valores através da engenharia reversa dos valores conhecidos apresentados em [25] e [26]. Para isso as 25 cores conhecidas (figura 4) entre 0 e 72 dbZ, em incrementos de 3 unidades foram extrapoladas, inserindo valores entre essas cores para representar as cores desconhecidas (figura 5).

Os valores das cores intermediárias foram calculados iterando entre as cores conhecidas e adicionando os valores da cor inicial com uma porcentagem da cor final (equação 4). Com isso também é possível definir os valores de refletividade correspondente a essas cores, tendo em vista que a cor intermediária deve ter valor menor que a cor inicial e menor que a cor final. Através desse método o gradiente de 24 cores, com incrementos de 3 unidades pode ser expandido para um gradiente de 73 cores com 1 unidade representada por cada cor.

Como algumas cores presentes nas imagens de radares não estão presentes no gradiente extrapolado, elas são aproximadas das cores conhecidas mais próximas visualmente através de uma conversão para o espaço de cores CIELAB, o qual permite a comparação perceptual de cores utilizando a distância euclidiana entre as mesmas [27].

Por fim, para converter os valores de radar coletados de dbZ para mm. Utilizando algumas informações em alguns artigos publicados pelo SIMEPAR podemos realizar algumas estimativas sobre os dados coletados. Os radares utilizados retornam valores entre 0dbZ e 72dbZ com base na reflexividade das partículas de precipitação. Ademais, o gradiente utilizado pelo software proprietário do SIMEPAR, RADEX, também possui algumas cores conhecidas, assim como seus correspondentes numéricos [26].

Para obter uma estimativa aproximada dos valores de precipitação, a imagem foi carregada e os valores de reflexividade foram aplicados a fórmula de Marshall-Palmer (3.3).

Algoritmo 3 Algoritmo para extrapolar cores e valores do gradiente

gradient ← [[73, 186, 74], [57, 176, 73], . . .] ▷ Gradiente contendo as cores (RGB) conhecidas)
gradient_values ← [0, 3, . . .] ▷ Valores de refletividade associados a cada cor no gradiente
new_gradient ← []
new_gradient_values ← []
$n\_subdivisions \leftarrow 2$
$i \leftarrow 0$
while $i < gradient.length - 1$ **do**
    $starting\_color \leftarrow gradient[i]$
    $ending\_color \leftarrow gradient[i + 1]$
    $start\_value \leftarrow gradient\_values[i]$
    $end\_value \leftarrow gradient\_values[i + 1]$
    new_gradient.append(starting_color)
    $j \leftarrow 0$
    while $j < n\_subdivisions$ **do**
        $percent \leftarrow (j + 1)/(n\_subdivisions + 1)$ ▷ Porcentagem da cor final a ser adicionada na nova cor
        new_color ← [$start[0]+percent*(end[0]-start[0])$, $start[1]+percent*(end[1]-start[1])$, $start[2] + percent * (end[2] - start[2])$] ▷ Calcula os valores dos canais RGB da nova cor
        $color\_value \leftarrow start\_value + \frac{end\_value-start\_value}{n\_subdivisions+1}(percent + 1)$ ▷ Valor de refletividade relacionado a cor intermediária
        new_gradient.append(new_color)
        new_gradient_values.append(color_value)
    end while
end while
new_gradient.append(gradient[gradient.length - 1]) ▷ Adiciona a última cor conhecida ao novo gradiente
new_gradient_values.append(gradient_values[gradient_values.length - 1])

Algoritmo 4 Algoritmo para aproximar cores desconhecidas da imagem para cores conhecidas do gradiente

min_dists ▷ Matriz contendo as menores distancias para cada pixel da imagem
for $k = 1$, $ki{+}{+}$, $k < gradient.length$
gradient_lab ← rgb2lab(gradient[k]) ▷ Converte a cor atual do gradiente para CIELAB
for $i = 1$, $i{+}{+}$, $i < image\_width$
for $j = 1$, $j{+}{+}$, $j < image\_height$
lab_color ← rgb2lab(image[ij]) ▷ Converte a cord atual da imagem para CIELAB
dist ← euclidian_distance(lab_color, gradient_lab)
if dist < mindists[ij] **then**
    $mindists[ij] \leftarrow dist$
    $image[ij] \leftarrow gradient[k]$ ▷ Cor no pixel ij da imagem é mais próxima da cor atual do gradiente
end if

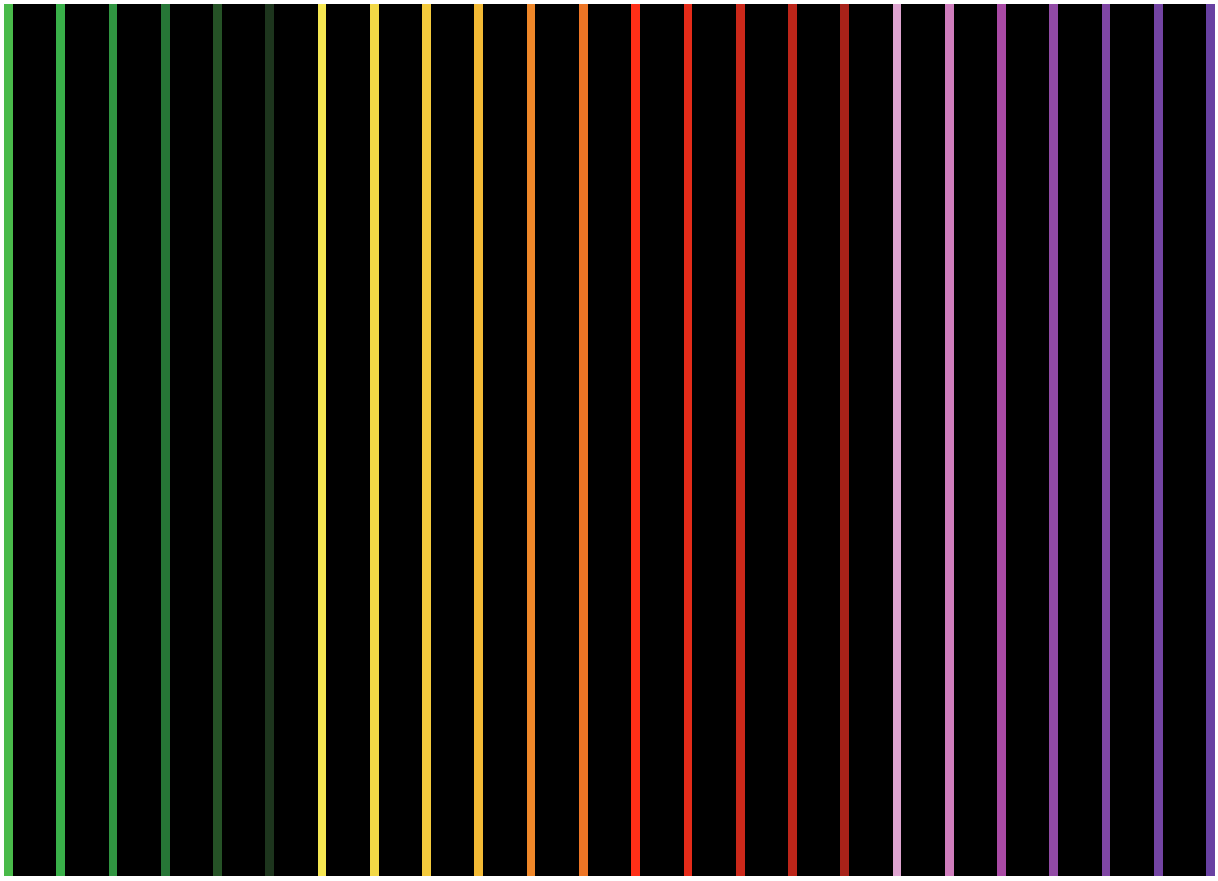


Figure 4 – Gradiente contendo cores com valores conhecidos no gradiente original

$$Z = aR^b \tag{3.3}$$

Onde:

Z é o valor do fator refletividade do radar Z ($\frac{mm^6}{mm^3}$)

R é o valor de precipitação em mm

a e b são coeficientes que podem variar entre localizações e entre estações

Como existem diversas variações na formula de Marshall-Palmer com diferentes valores para as constantes a e b, os valores comumente utilizados de $a = 200$ e $b = 1.6$ foram utilizados, tendo em vista que esta versão da formula apresenta resultados adequados em diversas partes do mundo e diferentes tipos de tempestades [26].

3.3 Armazenamento de Dados Espaço-Temporais

Para inserir os dados de radar no banco de dados foi utilizado a ferramente de linha de comando raster2pgsql, inclusa com o PostGIS e criada com base na biblioteca de georreferenciamento GDAL [20]. Isso permite que as 27247 imagens de radar sejam lidas, processadas e adicionadas com um único comando no banco de dados, ao mesmo tempo

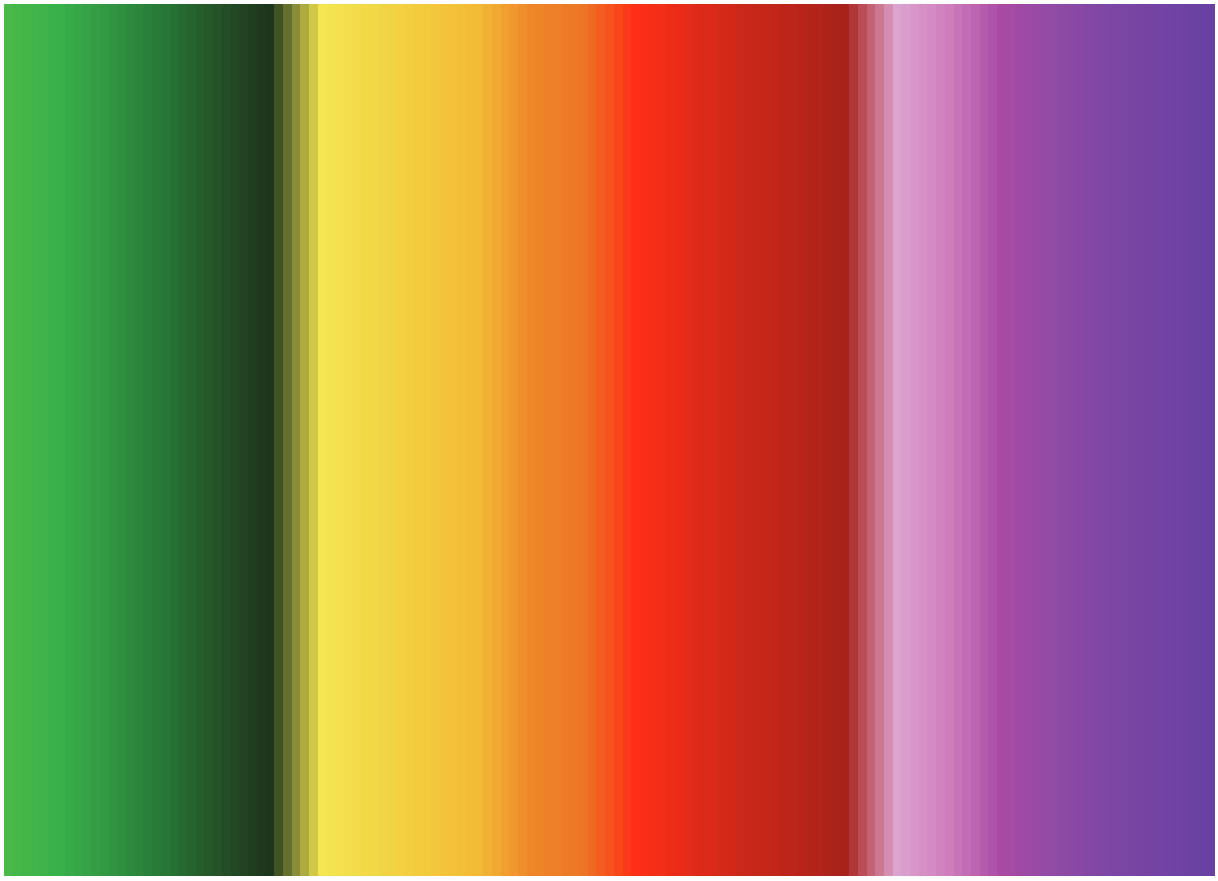


Figure 5 – Gradiente contendo as cores extrapoladas do gradiente

que propriedades de georreferenciamento são preservadas com o sistema de referencia espacial sendo detectado automaticamente e a relação entre pontos da imagem com valores de latitude e longitude sendo preservados.

Para a inserção dos dados o comando utilizado foi:

```
# raster2pgsql -s 4674 -F images/*.tiff radar_data | psql
```

Onde:

-s indica o sistema de referência espacial, no caso SIRGAS 2000

-F inclui o nome do arquivo raster na tabela

radar_data é o nome da tabela para inserção dos dados

A tabela final possuí as seguintes colunas:

Os dados inseridos são imagens raster, equivalentes a um mosaico de pixels, sendo representados por uma coluna de tipo raster baseado no tipo *bytea* na tabela Um raster pode ter múltiplas bandas de cores, as quais podem representar as mais diversas informações através dos valores associados a cada posição [20]. Imagens RGB, por exemplo, utilizam 3 bandas de cores, vermelho, verde, azul para representar uma imagem colorida.

Consulta 3.1 – Modelagem da tabela radar_data

```
CREATE TABLE "radar_data" (
        "rid" serial PRIMARY KEY,
        -- Arquivo raster carregado em formato binario
        "rast" raster,
        -- Nome do arquivo de radar
        "filename" text
);
```

É comum que dados raster sejam disponibilizados com mais frequência que a alternativa de dados vetoriais, tendo em vista dados raster tendem a ter origem em instrumentos de medição [20].

3.4 Consultas Sobre Dados Espaço-Temporais

Para a realização de consulta sobre os dados espaço-temporais inseridos anteriormente é possível combinar o poder expressivo da linguagem SQL disponibilizada pelo Postres com as várias funções para lidar com objetos espaciais disponibilizadas pela extensão PostGIS. Um exemplo disso é a forma com que os dados de banda dos raster armazenados são acessados, já que a tabela armazena o raster de forma binária e o valor da banda é acessado através da função ST_Value.

3.4.1 Listar metadados dos *rasters*

Na consulta 3.2 é apresentado a utilização da função ST_MetaData para a obter os metadados dos *rasters* armazenados na tabela radar_data. Útil para casos onde as informações de georreferenciamento, SRID, altura e largura do raster não são conhecidos ou podem apresentar diferenças entre si. Os resultados podem ser vistos na tabela 2.

Consulta 3.2 – Listar metadados de todos os *rasters*

```
SELECT
        rid,
        filename,
        (ST_MetaData(rast)).*
FROM
        radar_data;
```

Table 2 – Metadados dos *rasters*

rid	filename	upperleftx	upperlefty	width	height	scalex	scaley	skewx	skewy	srid	numbands
1	radar-2022-08-07-19-30.tiff	-57.151...	-21.008...	980	672	0.0115...	-0.011...	-4.411...	7.6612...	4674	1
2	radar-2022-08-07-19-40.tiff	-57.151...	-21.008...	980	672	0.0115...	-0.011...	-4.411...	7.6612...	4674	1
3	radar-2022-08-07-19-50.tiff	-57.151...	-21.008...	980	672	0.0115...	-0.011...	-4.411...	7.6612...	4674	1
4	radar-2022-08-07-20-00.tiff	-57.151...	-21.008...	980	672	0.0115...	-0.011...	-4.411...	7.6612...	4674	1
5	radar-2022-08-07-20-10.tiff	-57.151...	-21.008...	980	672	0.0115...	-0.011...	-4.411...	7.6612...	4674	1

Como todas os *rasters* na tabela 2 tem a mesma origem e seguem o mesmo processo de georreferenciamento as colunas possuem os mesmos valores.

3.4.2 Listar dados estatísticos dos *rasters*

A consulta 3.3 demonstra como os valores total, soma, média, desvio padrão, mínimo e máximo podem ser extraídos de cada raster. Um exemplo pode ser visto na tabela 3 onde 10 rasters foram listados e seus valores de precipitação analisados. Os resultados são apresentados na tabela 3.

Consulta 3.3 – Listar dados estatísticos dos *rasters*

```
SELECT
        rid,
        filename,
        (stats).*
FROM (
        SELECT
                rid,
                filename,
                ST_SummaryStats(rast, 1) AS stats
        FROM public.radar_data
) AS f
ORDER BY
        mean DESC;
```

3.4.3 Listar registros de precipitação no município de Londrina

Na consulta 3.4 é realizada uma consulta com o objetivo de listar todos os registros de precipitação que ocorreram na latitude (-51.1628) e longitude (-23.3045) do município de Londrina. Os valores em milímetros são retornado ordenados em ordem decrescente junto com o nome do arquivo de radar que contem esse registro.

Consulta 3.4 – Listar registros de precipitação no município de Londrina

```
SELECT
        filename,
        ST_Value(rast, ST_SetSRID(
        ST_MakePoint(-51.1628, -23.3045), 4674)
) AS pixel_value
FROM
        radar_data
WHERE
```

```
        ST_Intersects(rast, ST_SetSRID(
                ST_MakePoint(-51.1628, -23.3045), 4674)
        )
ORDER BY
        pixel_value DESC;
```

Foram utilizadas as funções espaciais para filtrar e acessar os pontos corretos do raster de acordo com a latitude e longitude definidas. A função ST_Makepoint tem como propósito definir um ponto contendo um par (latitude, longitude) que pode ser combinado com outras funções, contudo ainda é preciso definir o SRID do sistema de referenciamento espacial utilizado, devendo ser o mesmo que o utilizado ao georreferenciar as imagens, para este trabalho o SRID utilizado é 4674, o qual corresponde ao sistema de referencia espacial SIRGAS 2000.

Após o ponto ser criado a função ST_Intersects utiliza esse ponto para encontrar onde ele intersecciona o raster, considerando assim só os valores do município de Londrina.

Como o raster é armazenado em formato binário é preciso utilizar a função ST_Value para acessar o valor da única banda do raster, contendo nela o valor de precipitação. Novamente a função ST_Makepoint é utilizada para encontrar o ponto no raster referente a latitude e longitude do município de Londrina. Os resultados são apresentado na tabela 4.

3.4.4 Listar precipitação acumulada por hora no município de Londrina

Na consulta 3.5 é realizada o acumulo de precipitação horária para cada dia nos registros.

Para realizar essa consulta foi necessário obter o *timestamp* de captura da imagem, isso pode ser feito atráves do nome do arquivo, o qual sempre segue o formato: radar-$ano-$mes-$dia-$hora-$minuto.tiff. Com essa informação foi possível remover o prefixo 'radar-' e a extensão '.tiff', realizando o *parsing* dessa string para uma timestamp válida.

A função ST_Intersects é usada na condição WHERE para limitar a busca ao ponto correspondente a localização geográfica do município de Londrina. Os valores de precipitação para cada minuto é somado através da função SUM que recebe o resultado da função ST_Value como argumento. Este valor resultante da função ST_Value é o valor de precipitação no ponto com latitude e longitude igual a -51.1628 e -23.3045 respectivamente.

O resultado da consulta pode ser visto na tabela 5.

Table 3 – Dados estatísticos dos *rasters*

rid	filename	count	sum	mean	stddev	min	max
3461	radar-2022-10-06-10-10.tiff	658560	4589...	6.96...	12.99...	0	59
3463	radar-2022-10-06-10-30.tiff	658560	4524...	6.87...	12.84...	0	59
3462	radar-2022-10-06-10-20.tiff	658560	4523...	6.86...	12.86...	0	59
17062	radar-2023-02-02-20-30.tiff	658560	4511...	6.85...	12.42...	0	59
17066	radar-2023-02-02-21-10.tiff	658560	4501...	6.83...	12.39...	0	59
17067	radar-2023-02-02-21-20.tiff	658560	4492...	6.82...	12.33...	0	59
3465	radar-2022-10-06-10-50.tiff	658560	4489...	6.81...	12.86...	0	59
17065	radar-2023-02-02-21-00.tiff	658560	4486...	6.81...	12.40...	0	59
3466	radar-2022-10-06-11-00.tiff	658560	4468...	6.78...	12.89...	0	59
3464	radar-2022-10-06-10-40.tiff	658560	4465...	6.78...	12.78...	0	59

Table 4 – Registros de precipitação no município de Londrina

Filename	Pixel value
radar-2022-08-07-19-30.tiff	0
radar-2022-08-07-19-40.tiff	0
radar-2022-08-07-19-50.tiff	13
radar-2022-08-07-20-00.tiff	12
radar-2022-08-07-20-10.tiff	0
radar-2022-08-07-20-20.tiff	14
radar-2022-08-07-20-30.tiff	8
radar-2022-08-07-20-40.tiff	8
radar-2022-08-11-23-20.tiff	0
radar-2022-08-11-23-30.tiff	0

Consulta 3.5 – Listar precipitação acumulada por hora no município de Londrina

```
WITH space (londrina) AS (
        VALUES (
                ST_SetSRID(ST_MakePoint(-51.1628, -23.3045), 4674)
        )
)
SELECT
        EXTRACT(YEAR FROM timestamp) AS year,
        EXTRACT(MONTH FROM timestamp) AS month,
        EXTRACT(DAY FROM timestamp) AS day,
        EXTRACT(HOUR FROM timestamp) AS hour,
        SUM(
                ST_Value(rast, space.londrina)
        ) AS accumulated_hourly_precipitation
FROM
        public.radar_data, space
WHERE
        ST_Intersects(rast, space.londrina)
GROUP BY
        year, month, day, hour;
```

Consulta 3.6 – Listar precipitação acumulada por hora no município de Londrina entre 14/03/2023 16:00 e 14/03/2023 19:00 (inclusivo)

```
WITH space (londrina) AS (
        VALUES (
                ST_SetSRID(ST_MakePoint(-51.1628, -23.3045), 4674)
        )
)
SELECT
        EXTRACT(YEAR FROM timestamp) AS year,
        EXTRACT(MONTH FROM timestamp) AS month,
        EXTRACT(DAY FROM timestamp) AS day,
        EXTRACT(HOUR FROM timestamp) AS hour,
        SUM(
                ST_Value(rast, space.londrina)
        ) AS accumulated_hourly_precipitation
FROM
        public.radar_data, space
WHERE
        ST_Intersects(rast, space.londrina)
AND
        timestamp BETWEEN '2023-03-14 16:00:00'
                AND '2023-03-14 19:00:00'
GROUP BY
        year, month, day, hour;
```

3.4.5 Listar precipitação acumulada por hora no município de Londrina entre 14/03/2023 16:00 e 14/03/2023 19:00

A consulta 3.6 se baseia na consulta 3.5, porém possui duas condições adicionais. A data do evento de precipitação deve ser ao mesmo tempo maior ou igual a 14/03/2023 16:00 e menor ou igual a 14/03/2023 19:00.

O resultado da consulta pode ser visto na tabela 6.

Table 5 – Precipitação acumulada por hora no município de Londrina

year	month	day	hour	accumulated_hourly_precipitation
2022	8	7	19	13
2022	8	7	20	42
2022	8	11	23	0
2022	8	12	0	0
2022	8	15	19	0
2022	8	15	20	0
2022	8	15	21	0
2022	8	15	22	0
2022	8	15	23	0
2022	8	16	0	0

Table 6 – Precipitação acumulada por hora no município de Londrina entre 14/03/2023 16:00 e 14/03/2023 19:00 (inclusivo)

year	month	day	hour	accumulated_hourly_precipitation
2023	3	14	16	0
2023	3	14	17	0
2023	3	14	18	30
2023	3	14	19	15

4 RESULTADOS

4.1 Avaliação da Segmentação

Para avaliar de forma objetiva os resultados obtidos através da segmentação foi utilizado o processo sugerido em [15]. Onde cada imagem tem seu erro de classificação calculado, o qual pode ser definido como:

$$ME = \frac{|B_O \cap F_T| + |F_O \cap B_T|}{|B_O| + |F_O|} \tag{4.1}$$

Onde:

B_O são os pixels não considerados parte do objeto na imagem segmentada através de um algoritmo.

F_O são os pixels considerados parte do objeto na imagem segmentada através de um algoritmo.

B_T são os pixels não considerados parte do objeto na imagem segmentada manualmente.

F_T são os pixels considerados parte do objeto na imagem segmentada manualmente.

$|.|$ indica a cardinalidade do conjunto.

A partir dessa formula é possível definir o índice de similaridade n entre duas imagens e medir o quão próximo o resultado de um algoritmo é da imagem segmentada manualmente.

$$n = (1 - ME) * 100 \tag{4.2}$$

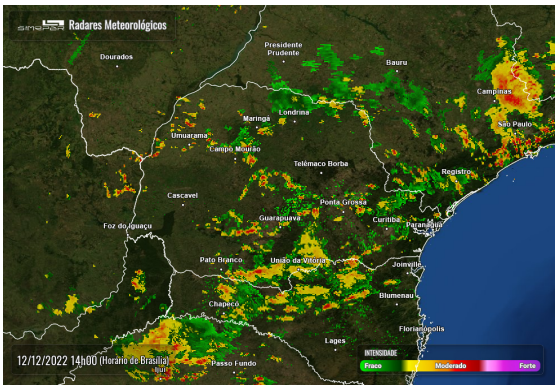
Quanto maior o valor de n, mais próximo está o resultado final da segmentação da imagem segmentada manualmente.

Foram registrados os valore de n referentes a 4 imagens selecionadas segmentadas utilizando algoritmos de evolução diferencial. Para efeitos de comparação também foram registrados os indicies de similaridade das imagens estudadas quando o algoritmo de Otsu foi aplicado a elas.

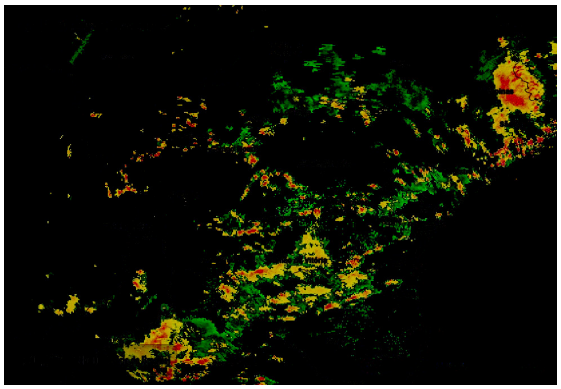
Como visto na tabela 7, ambos os algoritmos retornam resultados similares em comparação a imagem segmentada manualmente.

Figura	N_{Otsu}	N_{DE}
6	83,92%	84,33%
7	89,18%	89,39%
8	83,97%	83,76%
9	68,13%	68,09%

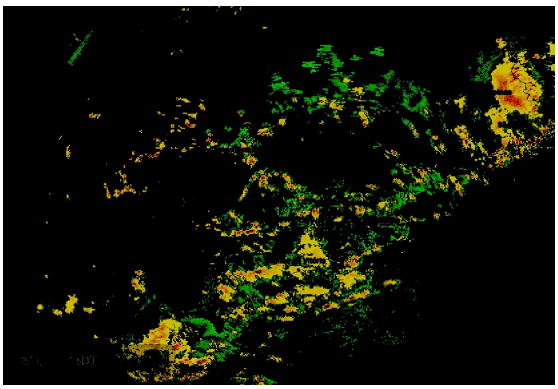
Table 7 – Índices de similaridade para diferentes figuras



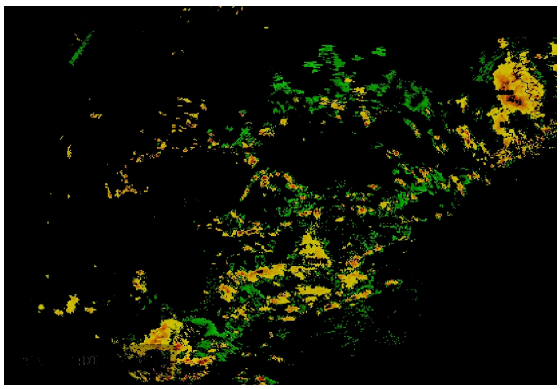
(a) Imagem de radar original



(b) Segmentação manual da imagem

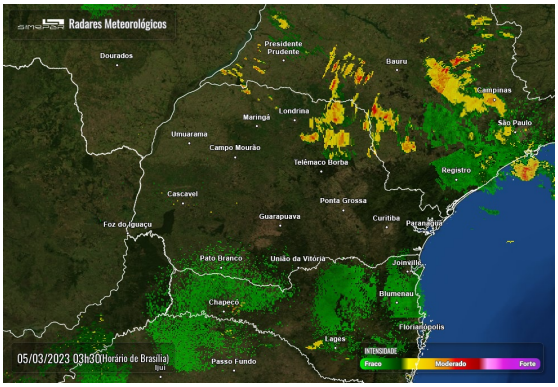


(c) Segmentação através de OTSU

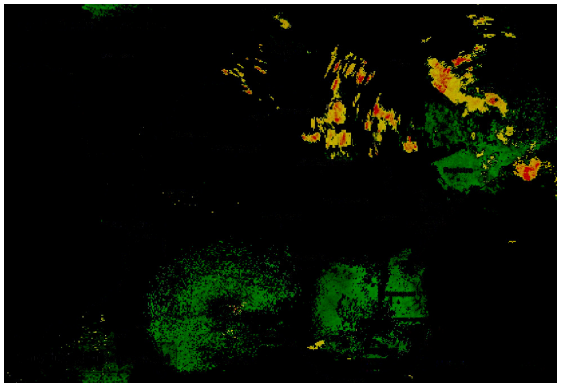


(d) Segmentação através de evolução diferencial

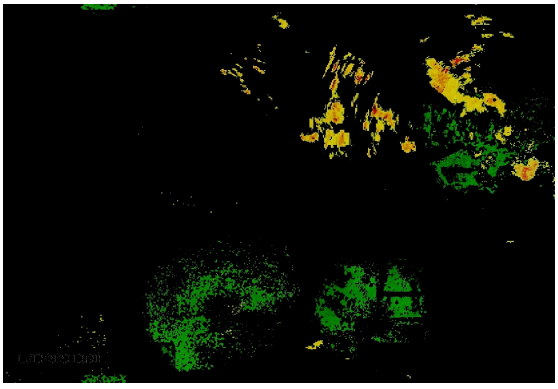
Figure 6 – Exemplo de segmentação 1



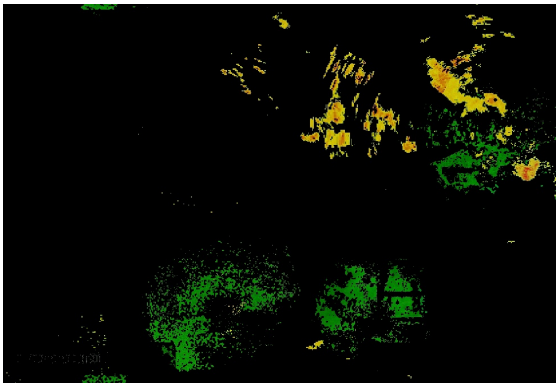
(a) Imagem de radar original



(b) Segmentação manual da imagem

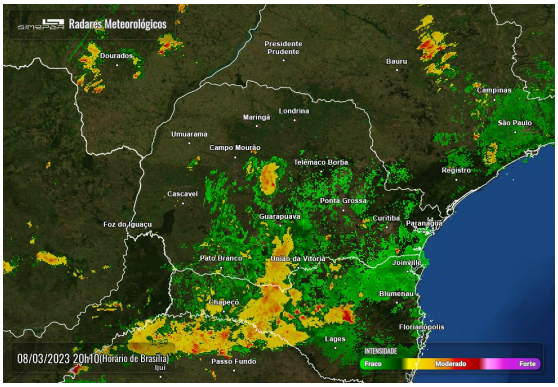


(c) Segmentação através de OTSU

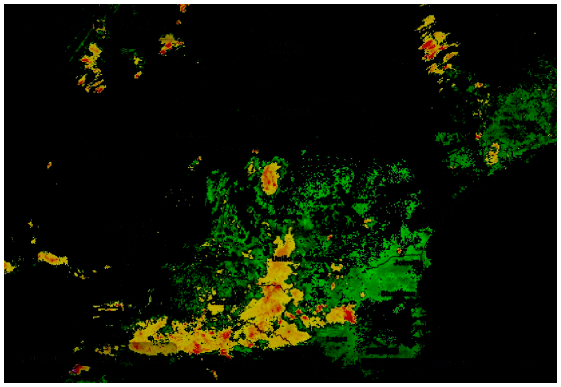


(d) Segmentação através de evolução diferencial

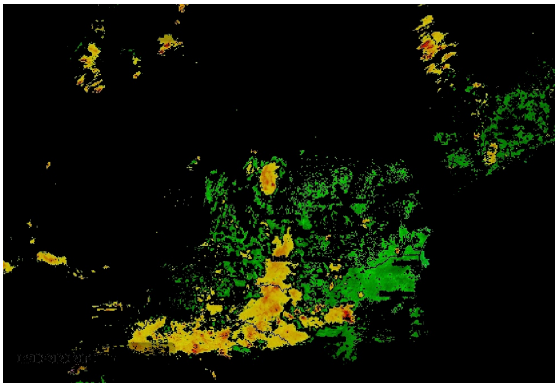
Figure 7 – Exemplo de segmentação 2



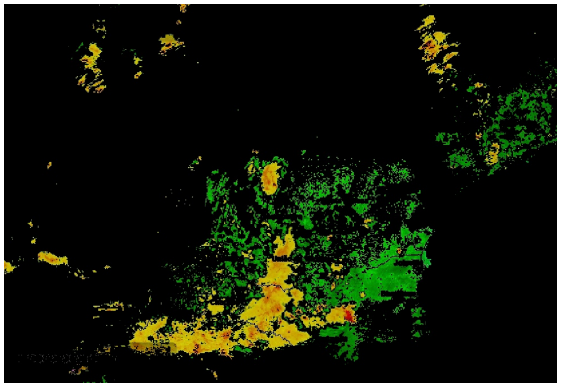
(a) Imagem de radar original



(b) Segmentação manual da imagem

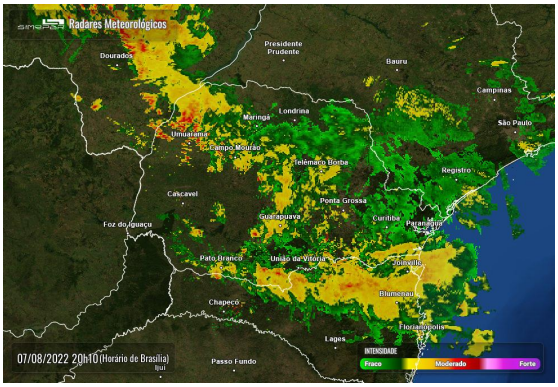


(c) Segmentação através de OTSU

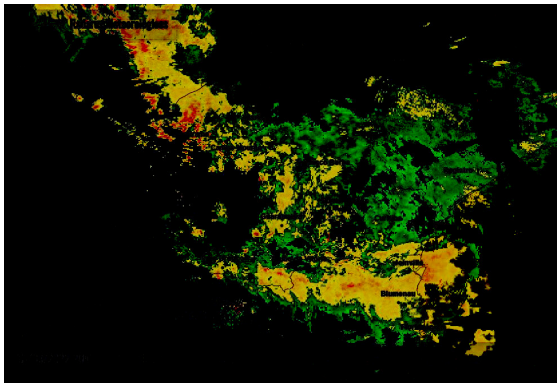


(d) Segmentação através de evolução diferencial

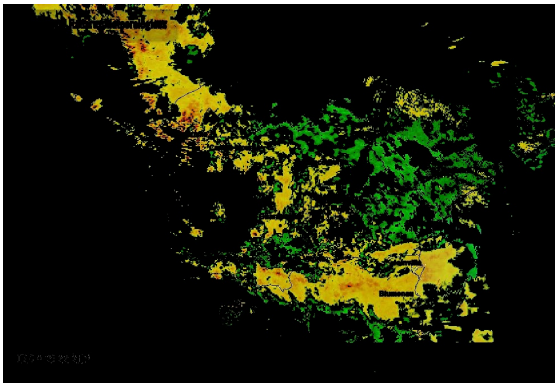
Figure 8 – Exemplo de segmentação 3



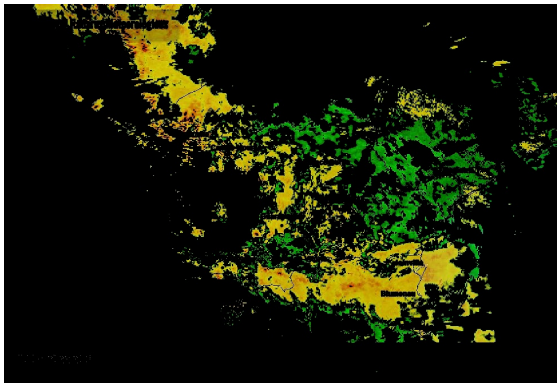
(a) Imagem de radar original



(b) Segmentação manual da imagem



(c) Segmentação através de OTSU



(d) Segmentação automática da imagem

Figure 9 – Exemplo de segmentação 4

4.2 Análise Quantitativa dos Dados

Buscando validar os valores numéricos obtidos 3.2.2, foram analisados os valores de precipitação acumulada, também disponibilizados pelo SIMEPAR, que permitem obter essa informação para cada estação no estado. Os dados de precipitação acumulada são disponibilizadas através de um arquivo JSON, contendo o valor de precipitação acumulado para cada hora do dia.

Para realizar a validação as imagens são agrupadas em períodos de 1 hora, para que assim possam ser comparadas diretamente com os dados de precipitação acumulada. Após as imagens do mesmo período serem determinas é necessário obter o valor de chuva acumulada em todas as 6 imagens nesse período, uma vez que o SIMEPAR disponibiliza novas imagens a cada 10 minutos.

Após repetir o algoritmo 5 para cada imagem foi possível obter a algumas observações sobre os dados (imagens 10, 11, 12, 13), sendo elas a média de erros entre o valor esperado de precipitação definido com base nas imagens de radares e os valores detectados nas estações meteorológicas.

Algoritmo 5 Algoritmo para validação dos valores de preciptação

$accumulated \leftarrow 0$
$minute \leftarrow 0$
while $minute < 6$ **do**
    $image \leftarrow load\_radar\_image(year, month, day, hour, minute)$ ▷ Carrega a imagem de radar na data especificada
    $rain\_around\_pixel \leftarrow images[ij]$ ▷ Obtêm a precipitação na posição especificada
    $accumulated \leftarrow accumulated + rain_around_pixel$
    $minute \leftarrow minute + 10$
end while
$ground\_truth \leftarrow load\_csv(year, month, day)$ ▷ Carrega o JSON com os dados de precipitação acumulada em cada estação
return abs(accumulated - ground_truth[hour]) ▷ Retorna a diferença na medição para a hora

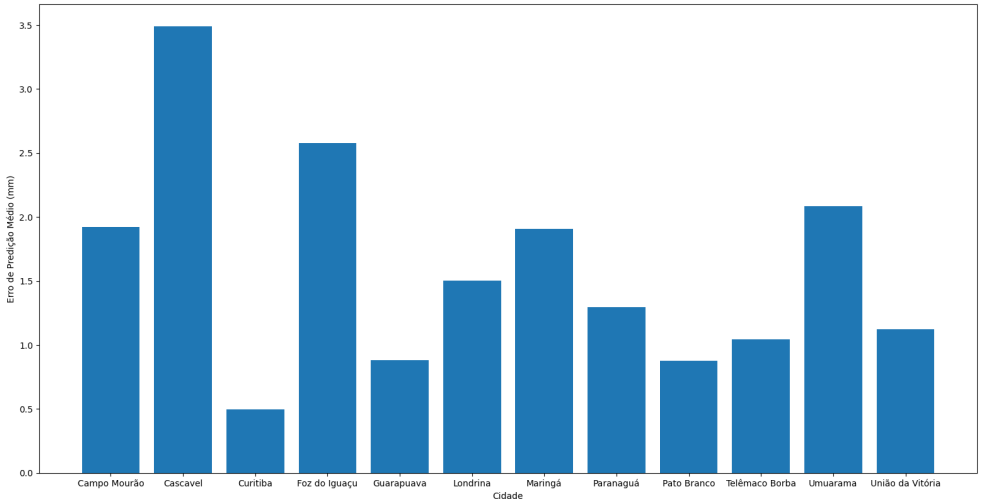


Figure 10 – Média de erros por cidade

4.3 Validação da Redução de Uso de Armazenamento de Dados

Para confirmar a redução no uso de armazenamento foram comparadas diferentes formas de armazenamento e seus respectivos tamanhos. Inicialmente, o conjunto de imagens geotiff possui 27247 imagens de radar, totalizando 46 gigas de armazenamento utilizado.

Entretanto, após a inserção dos *rasters* no banco de dados é possível notar uma grande redução no espaço utilizado. Utilizando a consulta 4.1 podemos verificar que os 46 gigas originais foram reduzidos a 1588 megabytes após a inserção dos mesmos no banco de dados. Todos os *rasters* utilizam o tipo 'Raster' disponibilizado pelo PostGIS, o qual é representado por um valor binário contendo todas as informações sobre o *raster* (banda, sistema geográfico, coordenadas, etc...) [20].

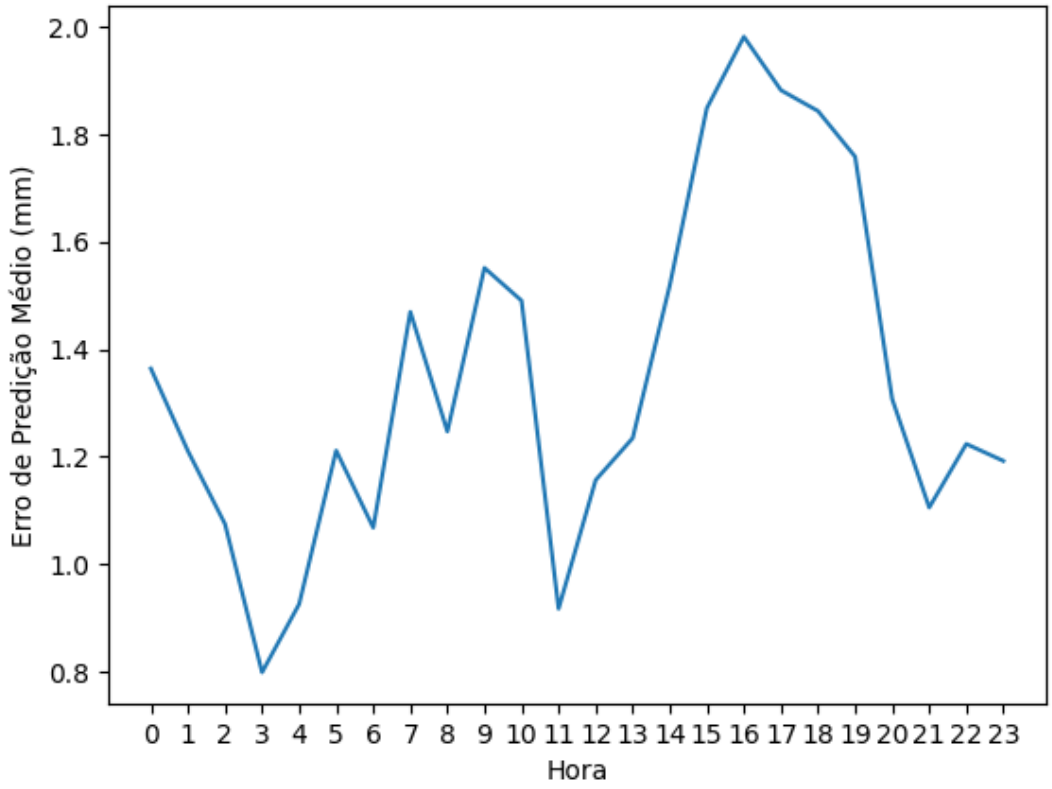


Figure 11 – Média de erros por hora

Consulta 4.1 – Obter o espaço em disco utilizado pela tabela radar_data

```
SELECT pg_size_pretty( pg_total_relation_size('radar_data') )
```

Outro ponto a ser considerado é o tamanho do conjunto de dados ao ser exportado para um arquivo para que outra pessoa possa reutilizar os dados já processados, evitando gastar tempo de processamento com as etapas de segmentação e categorização.

Para isso é possível utilizar o comando ogr2ogr da seguinte forma:

```
# ogr2ogr -f "GeoJSON"  radar_data.json PG:"host=myhost user=myuser dbname=mydatabase
password=mypassword" radar_data
```

Após os dados do banco de dados serem salvos no arquivo GeoJSON o mesmo utiliza 146 gigabytes de armazenamento, um resultado muito maior que todas as outras opções anteriores. Entretanto, analisando o arquivo é possível notar que 97% dos caracteres presentes são 0, uma vez que as os *rasters* armazenados pelo PostGIS em sua maioria também seguem o mesmo padrão.

Isso proporciona uma excelente oportunidade para a utilização de um algoritmo de

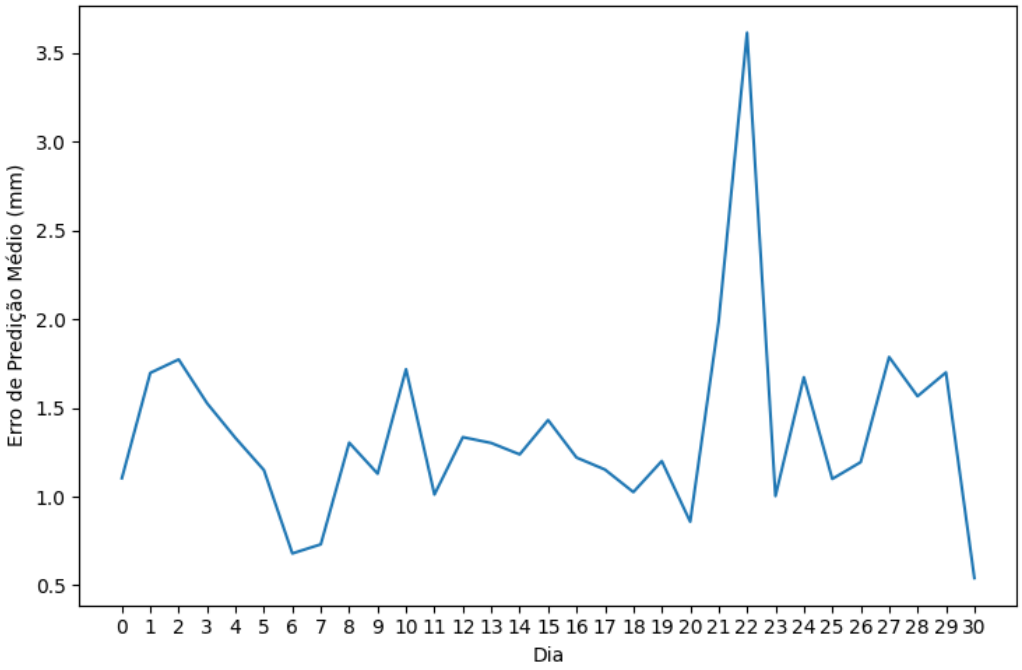


Figure 12 – Média de erros por dia

compressão capaz de notar esse padrão a fim de reduzir o tamanho utilizado pelo conjunto de dados exportado. Ao comprimir o arquivo com o programa zip houve um redução de 99% do tamanho do arquivo, onde o mesmo passou a utilizar apenas 846 megabytes para armazenar as 27247 imagens capturas. Com esse experimento é possível confirmar que o conjunto de dados criado é apropriado para ser disponibilizado para terceiros.

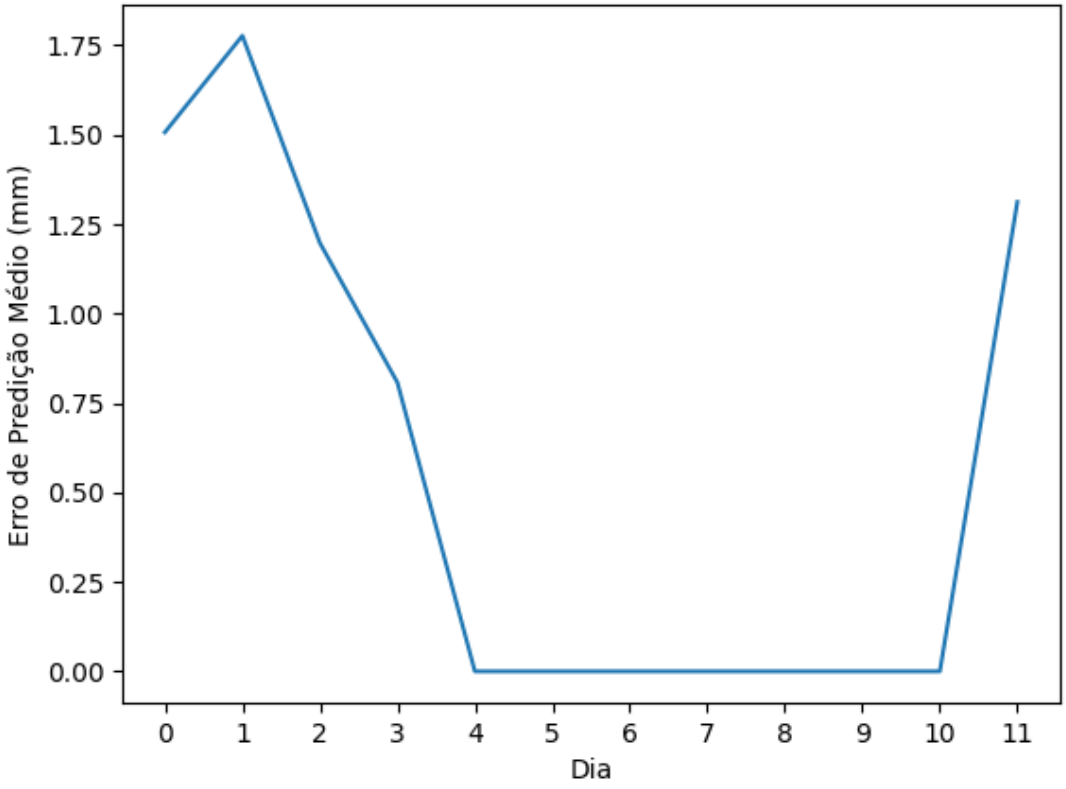


Figure 13 – Média de erros por mês (os meses de 4 à 10 não possuem dados)

5 CONCLUSÃO

Este trabalho apresentou o processo de criação de um conjunto de dados de precipitação para o estado do Paraná. Através das análises e processos realizados sobre as imagens de radar meteorológico disponibilizadas pelo SIMEPAR foi populado um banco de dados espaço-temporal totalizando 27247 imagens georreferenciadas e com valores de precipitação definidos.

Ao mesmo tempo, o armazenamento necessário para utilizar o conjunto de dados foi reduzido, ao mesmo tempo que o processo de utilização dos dados foi simplificado através de consultas SQL que podem expressar diversas buscas espaço-temporais.

Com isso, as principais contribuições deste trabalho são:

- Criação de um novo conjunto de dados de precipitação regional com base em tecnologias já estabelecidas.
- Avaliação quantitativa dos resultados apontados pelo conjunto de dados em relação a estações meteorológicas.
- Disponibilização de métodos para a reutilização do conjunto de dados por terceiros.

Todas essas contribuições são partes essenciais do conjunto de dados resultante, permitindo que terceiros utilizem os dados coletados, segmentados e georreferenciados de forma simples através de tecnologias existentes e já consolidadas, reaproveitando diversas otimizações presentes no Postgres e PostGIS. Isso resulta em um tempo de consulta curto para o usuário final, ao mesmo tempo que permite um uso mínimo de armazenamento em disco, quando comparado aos dados brutos, para a consulta de 27247 imagens de radar.

Como trabalhos futuros, é necessário analisar outros possíveis valores para as constantes a e b na fórmula de Marshall-Palmer (equação 3.3) em conjunto com um maior número de imagens de radar e de estações meteorológicas, visando encontrar os valores que melhor representem as chuvas no Paraná. Quanto maior o conjunto de testes disponível, mais abrangente e próximo da realidade serão os dados do conjunto de dados.

BIBLIOGRAPHY

[1] OLIVEIRA, A. C. Implementação do modelo atmosférico wrf acoplado com o modelo hidrológico topmodel para a bacia de união da vitória. *Programa de Pós Graduação em Engenharia de Recursos Hídricos e Ambiental do Setor de Tecnologia–Universidade Federal do Paraná. Curitiba*, 2006.

[2] TAPIADOR, F. J. et al. Global precipitation measurement: Methods, datasets and applications. *Atmospheric Research*, Elsevier, v. 104, p. 70–97, 2012.

[3] MITCHELL, T. D.; JONES, P. D. An improved method of constructing a database of monthly climate observations and associated high-resolution grids. *International Journal of Climatology: A Journal of the Royal Meteorological Society*, Wiley Online Library, v. 25, n. 6, p. 693–712, 2005.

[4] PARKER, M. *Digital Signal Processing 101, Second Edition: Everything You Need to Know to Get Started*. 2nd. ed. USA: Newnes, 2017. ISBN 0128114533.

[5] BENETI, C.; NOZU, I.; SARAIVA, E. Monitoramento da precipitação e de eventos de tempo severo com radar meteorológico no estado do paraná. p. 5, 1998.

[6] TURLAPATY, A. et al. Precipitation data merging using a machine learning based fusion. In: *EGU General Assembly Conference Abstracts*. [S.l.: s.n.], 2009. p. 4959.

[7] GAGNE, D. J.; MCGOVERN, A.; BROTZGE, J. Classification of convective areas using decision trees. *Journal of Atmospheric and Oceanic Technology*, v. 26, n. 7, p. 1341–1353, 2009.

[8] JITKAJORNWANICH, K. et al. Extracting storm-centric characteristics from raw rainfall data for storm analysis and mining. In: *Proceedings of the 1st ACM SIGSPATIAL international workshop on analytics for big geospatial data*. [S.l.: s.n.], 2012. p. 91–99.

[9] LIU, X.; WAN, Y. Storing spatio-temporal data in xml native database. In: IEEE. *2010 2nd International Workshop on Database Technology and Applications*. [S.l.], 2010. p. 1–4.

[10] XIE, H. et al. Gis-based nexrad stage iii precipitation database: automated approaches for data processing and visualization. *Computers & Geosciences*, Elsevier, v. 31, n. 1, p. 65–76, 2005.

[11] UDUPA, J. K. et al. A framework for evaluating image segmentation algorithms. *Computerized medical imaging and graphics*, Elsevier, v. 30, n. 2, p. 75–87, 2006.

[12] BANIMELHEM, O.; YAHYA, Y. A. Multi-thresholding image segmentation using genetic algorithm. In: THE STEERING COMMITTEE OF THE WORLD CONGRESS IN COMPUTER SCIENCE, COMPUTER . *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*. [S.l.], 2011. p. 1.

[13] PAL, N. R.; PAL, S. K. A review on image segmentation techniques. *Pattern recognition*, Elsevier, v. 26, n. 9, p. 1277–1294, 1993.

[14] CUEVAS, E. et al. Image segmentation based on differential evolution optimization. *Applications of evolutionary computation in image processing and pattern recognition*, Springer, p. 9–22, 2016.

[15] RAHNAMAYAN, S.; TIZHOOSH, H. R.; SALAMA, M. M. Image thresholding using differential evolution. In: *IPCV*. [S.l.: s.n.], 2006. p. 244–249.

[16] STORN, R.; PRICE, K. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, Springer Nature BV, v. 11, n. 4, p. 341, 1997.

[17] RAMADAS, M. et al. Segmentation of weather radar image based on hazard severity using rde: reconstructed mutation strategy for differential evolution algorithm. *Neural Computing and Applications*, Springer, v. 31, n. 2, p. 1253–1261, 2019.

[18] AL-AMRI, S. S.; KALYANKAR, N. V. et al. Image segmentation by using threshold techniques. *arXiv preprint arXiv:1005.4020*, 2010.

[19] BARTOSZEWSKI, D.; PIORKOWSKI, A.; LUPA, M. The comparison of processing efficiency of spatial data for postgis and mongodb databases. In: SPRINGER. *Beyond Databases, Architectures and Structures. Paving the Road to Smart Data Processing and Analysis: 15th International Conference, BDAS 2019, Ustroń, Poland, May 28–31, 2019, Proceedings 15*. [S.l.], 2019. p. 291–302.

[20] HSU, R. O. L. S. *PostGIS in Action, Third Edition*. 3. ed. [S.l.]: Manning, 2021. ISBN 1617296694; 9781617296697.

[21] MUPPIDI, M. et al. Image segmentation by multi-level thresholding using genetic algorithm with fuzzy entropy cost functions. In: IEEE. *2015 International conference on image processing theory, tools and applications (IPTA)*. [S.l.], 2015. p. 143–148.

[22] ASLANTAS, V.; TUNCKANAT, M. Differential evolution algorithm for segmentation of wound images. In: IEEE. *2007 IEEE International Symposium on Intelligent Signal Processing*. [S.l.], 2007. p. 1–5.

[23] YUAN, M. Representing complex geographic phenomena in gis. *Cartography and Geographic Information Science*, Taylor & Francis, v. 28, n. 2, p. 83–96, 2001.

[24] URBANO, F.; CAGNACCI, F. *Spatial database for GPS wildlife tracking data: a practical guide to creating a data management system with PostgreSQL/PostGIS and R*. [S.l.]: Springer Science & Business Media, 2014.

[25] CALVETTI, L.; FILHO, A. J. P. Ensemble hydrometeorological forecasts using wrf hourly qpf and topmodel for a middle watershed. *Advances in Meteorology*, Hindawi, v. 2014, 2014.

[26] SELUZNIAK, R. H. L. et al. Proposta de algoritmo integrado a sistema interativo para controle de qualidade e visualização de dados de radares meteorológicos.

[27] ANILKUMAR, K.; MANOJ, V.; SAGI, T. Colour based image segmentation for automated detection of leukaemia: a comparison between cielab and cmyk colour spaces. In: IEEE. *2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET)*. [S.l.], 2018. p. 1–6.