UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

Gilberto Perello Ricci Neto

Classificação e Rastreamento de Nuvens no Horizonte para Aplicações Nowcasting via Segmentação Semântica e Estimativa de Fluxo Ótico

Florianópolis
2023

Gilberto Perello Ricci Neto

Classificação e Rastreamento de Nuvens no Horizonte para Aplicações Nowcasting via Segmentação Semântica e Estimativa de Fluxo Ótico

Trabalho de Conclusão de Curso do Curso de Graduação em Sistemas de Informação do Centro Tecnológico da Universidade Federal de Santa Catarina para a obtenção do título de bacharel em Sistemas de Informação.
Orientador: Prof. Aldo Von Wangenheim , Dr.
Coorientadora: Juliana Marian Arrais , M.a.

Florianópolis
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Neto, Gilberto Perello Ricci
Classificação e Rastreamento de Nuvens no Horizonte para Aplicações Nowcasting via Segmentação Semântica e Estimativa de Fluxo Ótico / Gilberto Perello Ricci Neto ; orientador, Aldo von Wangenheim, coorientadora, Juliana Marian Arrais, 2023.
74 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Sistemas de Informação, Florianópolis, 2023.

Inclui referências.

1. Sistemas de Informação. 2. Segmentação Semântica. 3. Fluxo Óptico. 4. Nowcasting. 5. Nuvens. I. von Wangenheim, Aldo. II. Arrais, Juliana Marian. III. Universidade Federal de Santa Catarina. Graduação em Sistemas de Informação. IV. Título.

Gilberto Perello Ricci Neto

Classificação e Rastreamento de Nuvens no Horizonte para Aplicações Nowcasting via Segmentação Semântica e Estimativa de Fluxo Ótico

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "bacharel em Sistemas de Informação" e aprovado em sua forma final pelo Curso de Graduação em Sistemas de Informação.

Florianópolis, 8 de dezembro de 2023.

Prof. Renato Cislaghi, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Aldo Von Wangenheim , Dr.
Orientador

Juliana Marian Arrais, M.a.
Avaliadora
Universidade Federal de Santa Catarina

Prof. Sylvio Luiz Mantelli Neto, Dr.
Avaliador
Instituto Nacional de Pesquisas Espaciais - INPE

RESUMO

A influência das condições climáticas na incidência de irradiação solar representa um desafio à produção de energia fotovoltaica. Nuvens movendo-se ao redor do sol podem provocar saltos na irradiância em questão de segundos, resultando não apenas em perda de eficiência na produção de energia, como também em danos aos equipamentos. Para prever o impacto das nuvens em sistemas fotovoltaicos, métodos de previsão solar intra-hora (*nowcasting*) têm sido propostos. O presente trabalho utiliza técnicas de *deep learning* e de fluxo óptico para previsão intra-hora dos efeitos das nuvens. Esta previsão consiste na identificação de nuvens em imagens do céu, seguida pelo rastreio de sua movimentação. A identificação é feita via segmentação semântica, usando o modelo de rede neural PP-LiteSeg, treinado com um *dataset* de imagens de nuvens no horizonte. Detalhes sobre a criação deste *dataset* também são descrito no presente trabalho. Por sua vez, no rastreamento do movimento das nuvens, é empregado o método de fluxo óptico desenvolvido por Gunnar-Farneback. A predição é feita através de programa desenvolvido em Python, que processa uma sequência de fotos do horizonte e produz um vídeo indicando o tipo de cada nuvem, e também a direção e a velocidade de seu movimento. O programa classifica as nuvens com uma acurácia de 0,8698 e aponta, de maneira precisa, a direção de movimento através de setas inseridas nas imagens resultantes. Os resultados alcançados no presente trabalho indicam um grande potencial da técnica desenvolvida na área de energias renováveis.

Palavras-chave: segmentação semântica; fluxo óptico; nuvens; *nowcasting*; intra-hora; fotovoltaica.

ABSTRACT

The influence of weather conditions on solar irradiance poses a challenge to photovoltaic energy production. Clouds moving around the sun can cause abrupt fluctuations in irradiance within seconds, leading to not only a loss of efficiency in energy production but also potential damage to equipment. To forecast the impact of clouds on photovoltaic systems, nowcasting methods have been proposed. The present work employs deep learning and optical flow techniques for intra-hour prediction of cloud effects. This prediction involves cloud identification in sky images, followed by tracking their movement. Identification is performed through semantic segmentation using the PP-LiteSeg neural network model, trained on a dataset of horizon cloud images. Details regarding the creation of this dataset are also described in this work. For cloud motion tracking, the optical flow method developed by Gunnar-Farneback is employed. Prediction is carried out through a Python program that processes a sequence of horizon photos, producing a video indicating the type of each cloud, along with its direction and speed of movement. The program classifies clouds with an accuracy of 0.8698 and accurately indicates the direction of movement through arrows inserted in the resulting images. The results achieved in this study suggest significant potential for the developed technique in the field of renewable energy.

Keywords: semantic segmentation; optical flow; clouds; nowcasting; intra-hour; photovoltaic.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A necessidade de reduzir a dependência de matrizes energéticas poluentes em esgotamento, como os combustíveis fósseis, na produção de energia elétrica, vêm popularizando a adoção de tecnologias que usam fontes renováveis de energia (Kumari; Toshniwal, 2021). Dentre os vários tipos de fonte de energia sustentáveis, a energia solar se destaca (Kumari; Toshniwal, 2021), devido a fatores como o declínio do custo dos equipamentos de geração de energia fotovoltaica (Nascimento; Souza Viana *et al.*, 2019) e a abundância de energia solar no planeta. A irradiância solar média recebida pela superfície da Terra, se convertida totalmente em eletricidade, seria capaz de atender a demanda energética de todos os países do mundo (Kumari; Toshniwal, 2021).

Atualmente, é possível gerar eletricidade a partir da energia solar de maneira bastante eficiente, de modo que sistemas fotovoltaicos conectados à rede elétrica apresentam resposta quase linear à irradiância solar (Nascimento; Souza Viana *et al.*, 2019). Entretanto, a incidência de irradiância solar é altamente influenciada por condições climáticas (Martins; Neto; Rüther, 2022), de modo que sua natureza intermitente é um obstáculo para produção de energia fotovoltaica de maneira estável. Nuvens movendo-se na frente ou ao redor do sol são capazes de produzir saltos na irradiância (Nascimento; Braga *et al.*, 2020), de modo que a passagem de nuvens sobre coletores de energia solar pode reduzir sua potência de saída em 80% em questão de segundos (Hill *et al.*, 2012). Além disso, em determinadas circunstâncias, a presença de nuvens, que normalmente diminui a intensidade da irradiação solar sobre a superfície terrestre, pode também aumentá-la, causando um fenômeno conhecido como sobreirradiância. Esse fenômeno ocorre quando nuvens alinham-se ao sol em ângulos específicos, causando reflexões múltiplas de luz solar em uma única área e, consequentemente, aumentando a irradiação nesta região (Martins; Neto; Rüther, 2022).

A intermitência intrínseca da irradiação solar sobre painéis fotovoltaicos pode trazer diversos problemas na produção de energia. Um deles é a baixa eficiência na geração de eletricidade, já que nem todas as tecnologias fotovoltaicas conseguem manter eficiência nominal em períodos de irradiação fraca (Nascimento; Braga *et al.*, 2020). Quando tais sistemas estão ligados à rede elétrica, para que seja possível manter o nível exigido de fornecimento de energia durante os períodos de baixa irradiação solar, acabam sendo necessários geradores de energia complementares, o que aumenta o custo de tais sistemas (Inman; Pedro; Coimbra, 2013). Além disso, a oscilação na irradiância têm como consequência uma oscilação de voltagem, causando problemas em outros equipamentos da rede, como capacitores chaveados, em reguladores de tensão de carga e mudança de fase (Hill *et al.*, 2012), e também pode reduzir o tempo de vida de baterias de armazenamento (Nascimento; Braga *et al.*, 2020). Por sua vez, a sobreirradiância pode causar problemas como sobrecarga de inversores, queima de fusíveis e desligamentos em cascata de circuitos

de proteção em grandes usinas fotovoltaicas (Martins; Neto; Rüther, 2022).

Neste cenário onde a volatilidade da produção de energia fotovoltaica é um entrave, formas de antecipar a quantidade de irradiação que incidirá sobre os painéis solares são desejáveis, e uma forma eficaz de fazê-lo é empregar técnicas de previsão solar. Como as oscilações na irradiância podem ocorrer em questão de segundos, as técnicas com o menor horizonte de predição são as mais adequadas para a tarefa. Assim, diversos métodos de *nowcasting* (previsão intra-hora), capazes de antecipar irradiação de minutos à frente, vêm sido propostos para melhorar o desempenho de plantas fotovoltaicas (Lin; Zhang; Wang, 2023).

A revisão sistemática de literatura de Lin, Zhang e Wang (2023) categoriza as técnicas de previsão solar intra-hora de acordo com o tipo de informação que usam para predizer a irradiância ou a potência de saída dos painéis solares. Desta forma, os métodos *image-based* seriam o que usam imagens de nuvens no céu para fazer predições, enquanto os métodos *data-driven* seriam os que usam dados meteorológicos para tal. Lin, Zhang e Wang (2023) argumentam que os métodos *image-based* teriam uma vantagem natural sobre os *data-driven*, já que as imagens contém informações mais intuitivas a respeito da movimentação e mudanças nas nuvens. Devido a sua maior resolução temporal e espacial, os métodos que usam imagens de câmeras terrestres apresentam vantagens sobre os demais métodos *image-based*, como aqueles que utilizam imagens de satélites, fornecendo maiores detalhes sobre o local de interesse e detectando mudanças mais sutis no posicionamento das nuvens (Lin; Zhang; Wang, 2023).

Na maioria das soluções que usam câmeras terrestres para a captura de imagens, emprega-se equipamento que usa lentes olho-de-peixe, apontadas para o zênite, para a realização das predições (Arrais *et al.*, 2022). Câmeras deste tipo, além de terem custo elevado (Martins; Cerentini *et al.*, 2022), promovem distorções nas imagens devido ao formato redondo de sua lente, o que torna necessária uma etapa de pré-processamento para correção (Arrais *et al.*, 2022). Através do uso de imagens do horizonte, que podem ser capturadas com câmeras convencionais, seria possível contornar os problemas relacionados às câmeras olho-de-peixe (Arrais *et al.*, 2022).

As imagens de nuvens podem ser utilizadas de duas formas diferentes na previsão solar. Pode-se efetuar as predições diretamente a partir das imagens, ou pode-se usar informações sobre a movimentação das nuvens no processo preditivo (Lin; Zhang; Wang, 2023). Esta última abordagem costuma ser a mais utilizada, pois, apesar de exigir uma etapa de rastreio de movimento de nuvens, costuma aumentar significativamente a acurácia da previsão (Lin; Zhang; Wang, 2023).

O processo de rastreamento de movimento de nuvens em imagens do céu normalmente passa por uma etapa de identificação das nuvens, identificação esta que costuma ser tratada como um problema de segmentação semântica (Lin; Zhang; Wang, 2023). O formato irregular e a fragmentação das nuvens (Lin; Zhang; Wang, 2023), aliados ao fato de

que atividades de observação sinóptica são variáveis e subjetivas, feitas majoritariamente por seres humanos (Neto *et al.*, 2022), indicam que a criação de critérios objetivos para identificar nuvens pode envolver uma quantidade muito grande de características para classificação, o que tornaria vantajoso o uso de *deep learning* (Mahony *et al.*, 2019). Entretanto, trabalhos na área dificilmente adotam técnicas de *machine learning*, empregando, em sua maioria, métodos clássicos para a tarefa de identificação (Arrais *et al.*, 2022).

No presente trabalho, é proposta uma solução para classificação e rastreamento de nuvens em imagens do céu, com aplicação em previsão solar intra-hora. O processo envolve a segmentação semântica das nuvens e a estimativa de seu deslocamento. Afim de explorar as possíveis vantagens do uso de *deep learning* na classificação de nuvens, empregou-se um modelo de rede neural no estado-da-arte, o PP-LiteSeg, para a realização da segmentação semântica. A rede foi treinada com um *dataset* de imagens de nuvens no horizonte, abordagem esta que evita problemas associados ao uso de câmeras olho-de-peixe. Já a predição de movimento foi feita através do método de fluxo óptico denso de Gunnar-Farneback, que, diferente de métodos de fluxo óptico esparso, não depende de *pixels*-chave, difíceis de serem identificados em nuvens, para o cálculo de deslocamento. A solução, implementada em Python, recebe uma sequência temporal de imagens de nuvens no horizonte como entrada e, como saída, retorna um vídeo com segmentação e indicação de movimento das nuvens.

1.1 OBJETIVOS

Nas subseções a seguir, serão apresentados os objetivos gerais e específicos do presente trabalho.

1.1.1 Objetivo principal

O objetivo principal do trabalho é demonstrar a viabilidade da realização de previsões intra-hora da movimentação de nuvens no céu, com aplicação na área de produção de energia solar, através de redes neurais treinadas com imagens do horizonte, onde se detecta e classifica as nuvens através de segmentação semântica e rastreia-se a direção de seu movimento através de estimativas de fluxo ótico.

1.1.2 Objetivos específicos

Os objetivos específicos são:

- Treinar o modelo de rede neural PP-LiteSeg para segmentação semântica de nuvens no horizonte;
- Usar o método de estimativa de fluxo ótico denso de Gunnar-Farneback para prever o movimento das nuvens;

- Afim de demonstrar a factibilidade do uso das tecnologias mencionadas, implementar solução experimental que receba uma sequência temporal de imagens de nuvens no horizonte como entrada e, como saída, retorne um vídeo com segmentação e indicação de movimento das nuvens.

1.2 ESTRUTURA DO TRABALHO

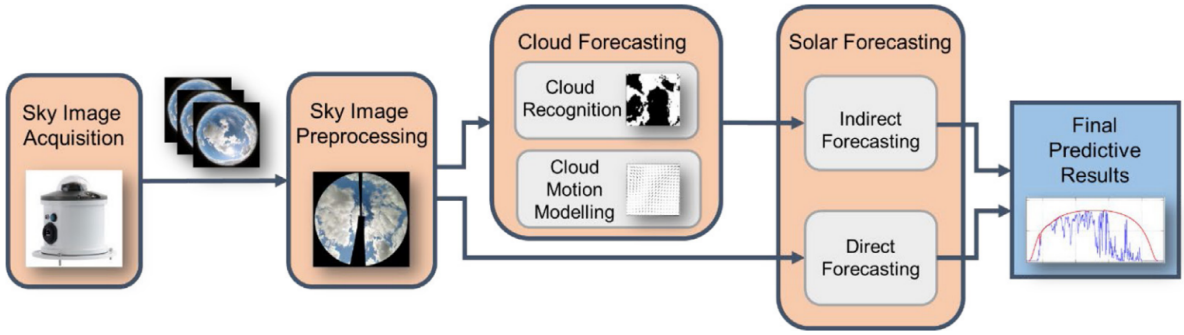
No próximo capítulo, REVISÃO BIBLIOGRÁFICA, será apresentado um panorama de como trabalhos recentes têm abordado o problema da previsão da irradiação solar a partir de imagens do céu. Em seguida, no capítulo METODOLOGIA, detalharemos tanto o conjunto de dados utilizado para o treinamento da rede neural quanto o desenvolvimento da solução proposta, incluindo o segmentador semântico, o rastreador de fluxo óptico e a criação dos vídeos. No capítulo RESULTADOS E DISCUSSÃO, avaliaremos o desempenho do modelo de segmentação semântica e apresentaremos uma análise qualitativa dos vídeos gerados pela solução desenvolvida. Por fim, nos capítulos CONCLUSÃO e RECOMENDAÇÕES PARA TRABALHOS FUTUROS, serão apresentadas, respectivamente, considerações finais sobre o trabalho e indicações de melhorias para a solução proposta a serem exploradas em trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

A análise de trabalhos relacionados aqui apresentada é feita a partir de síntese das revisões sistemáticas de literatura de Lin, Zhang e Wang (2023) e Arrais *et al.* (2022). O trabalho de Lin, Zhang e Wang (2023) investiga as principais abordagens utilizadas para previsão solar intra-hora baseados em imagens do céu capturadas a partir do solo, propondo um *framework* genérico para descrevê-las. Já em Arrais *et al.* (2022), temos a análise de 32 artigos, publicados entre 2013 e o primeiro semestre de 2022, que tratam de rastreamento de nuvens a partir de imagens do céu capturadas do solo aplicados à previsão solar de curto prazo.

O trabalho de Lin, Zhang e Wang (2023) parte do entendimento de que soluções que usam imagens para a previsão solar intra-hora passar por um conjunto comum de fases na realização de suas tarefas. Estas tarefas comuns são apresentadas através de um *framework*, cujo diagrama pode ser visto na Figura 1. A descrição de cada um dos módulos do *framework* pode ser vista a seguir:

Figura 1 – Módulos do *framework* proposto por Lin, Zhang e Wang (2023).



Fonte: Lin, Zhang e Wang (2023).

- Módulo 1 - *Sky Image Acquisition* (Aquisição de Imagens do Céu): módulo onde se realiza a captura de imagens do céu, através de algum tipo de câmera;
- Módulo 2 - *Sky Image Preprocessing* (Pré-processamento de Imagens do Céu): módulo que aplica transformações nas imagens obtidas, com o intuito de deixá-las em formato adequado para a execução dos algoritmos de predição;
- Módulo 3 - *Cloud Forecasting* (Previsão de Nuvens): módulo opcional, dividido em dois submódulos, *Cloud Recognition* (Reconhecimento de Nuvens) e *Cloud motion modeling* (Modelagem de Movimento de Nuvens), e onde é feita identificação e rastreamento de nuvens;

- Módulo 4 - *Solar Forecasting* (Previsão Solar): módulo onde, a partir da imagem pré-processada ou da predição do movimento das nuvem, são calculados os resultados preditivos finais.

Devido ao fato de a classificação e predição de movimento de nuvens, questões centrais do presente trabalho, corresponderem, respectivamente, aos submódulos de Previsão de Nuvens e de Modelagem de Movimento de Nuvens do *framework* proposto por Lin, Zhang e Wang (2023), adotaremos esta perspectiva durante o estudo das abordagens tecnológicas utilizadas em pesquisas relacionados.

As seções a seguir correspondem ao Módulo 1, de Aquisição de Imagens do Céu e ao Módulo 3, de Previsão de Nuvens, de modo que esta última seção é dividida em subseções correspondentes aos submódulos de Reconhecimento de Nuvens e de Modelagem de Movimentação de Nuvens. Optou-se por esta subdivisão devido ao entendimento de que estas são etapas contempladas pela solução proposta no presente trabalho. Cada seção traz as tecnologias empregadas nos estudos correlatos para implementação do módulo correspondente. A análise quantitativa das técnicas utilizadas usa estatísticas trazidas pela revisão sistemática de Arrais *et al.* (2022).

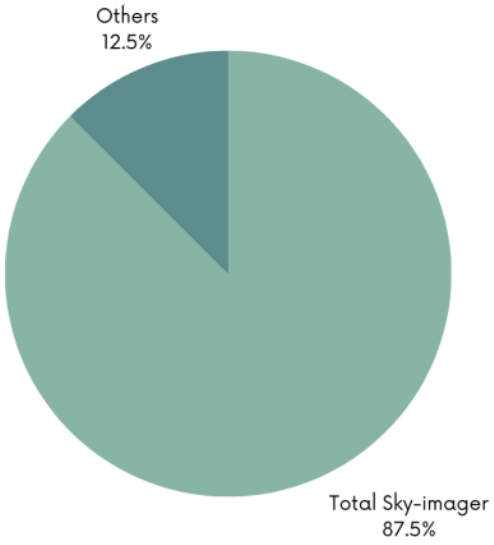
2.1 AQUISIÇÃO DE IMAGENS DO CÉU

A grande maioria dos trabalhos investigados em Arrais *et al.* (2022) empregou algum tipo de câmera semi-profissional com lente olho-de-peixe para a captura de imagens do céu. De acordo com o apresentado na Figura 2, 87.5% dos artigos usaram instrumentos do tipo TSI *Total Sky Imager*, que estão entre os dispositivos equipados com este tipo de lente.

Conforme exposto na introdução do presente trabalho, câmeras olho-de-peixe distorcem a imagem capturada, fazendo com que ela perca características importantes para o seu uso. Esta distorção ocorre devido ao processo de formação de imagem da câmera, que envolve a projeção linear de pontos do espaço real/tridimensional do céu em uma esfera virtual, que é seguida da projeção não-linear dos pontos desta esfera em uma imagem plana (Lin; Zhang; Wang, 2023). Esta não-linearidade provoca uma distorção radial na imagem (Lin; Zhang; Wang, 2023). Além disso, o fato de as lentes olho-de-peixe apontarem para o zênite faz com que não seja possível fazer o cálculo da altura da base de nuvem. Isto, por sua vez, impede a classificação da nuvem de acordo com os critérios da *World Meteorological Organization* (WMO) (Arrais *et al.*, 2022).

Além da perda de informações valiosas, o uso de câmeras olho-de-peixe pode tornar necessária uma fase de pré-processamento de imagens antes que um algoritmo de previsão de nuvens possa utilizá-las (Lin; Zhang; Wang, 2023). Isto ocorre porque muitos desses algoritmos exigem como entrada imagens sem distorções, capturadas com câmera padrão (Lin; Zhang; Wang, 2023). O presente trabalho não usa imagens capturadas com lentes olho-

Figura 2 – Tipos de câmeras usadas na aquisição de imagens do céu em diferentes trabalhos.



Fonte: Arrais *et al.* (2022).

de-peixe, evitando assim o problema da distorção e a necessidade de pré-processamento de imagens.

2.2 PREVISÃO DE NUVENS

2.2.1 Reconhecimento de nuvens

Assim como o presente trabalho, os artigos analisados por Lin, Zhang e Wang (2023) tratam o problema de reconhecimento de nuvens como sendo um problema de segmentação semântica em Visão Computacional. Algoritmos de segmentação semântica aplicados ao reconhecimento de nuvens, além de segmentar as nuvens de forma acurada, devem ser capazes de identificar quais *pixels* ou regiões da imagem correspondem ao céu e quais correspondem à nuvens, muitas vezes sendo necessário distinguir a qual tipo de nuvem pertence determinado *pixel*. Outro desafio na implementação destes algoritmos é a velocidade da segmentação, que deve ser adequada para *nowcasting* (Lin; Zhang; Wang, 2023).

Características específicas das imagens do céu fazem com que a tarefa de detectar e classificar nuvens não seja resolvida facilmente por métodos convencionais de segmentação. Lin, Zhang e Wang (2023) argumenta que a distribuição de nuvens no céu é irregular, fragmentada e sem forma fixa, além de existirem problemas como a saturação de luminosidade próxima ao disco solar. Por sua vez, Arrais *et al.* (2022) menciona o problema da esparsidade do limite das nuvens, que torna difícil a detecção de suas bordas, e também o problema da mudança de brilho devido a variações climáticas, que provoca ruídos e e

instabilidade no contraste de cores da imagem.

Lin, Zhang e Wang (2023) divide as abordagens para reconhecimento de nuvens em:

- *Thresholding* (Limiarização), onde se traça um ou mais limites, baseados em informações de cores, limite(s) este(s) usados para dividir os *pixels* de uma imagem em duas ou mais classes (como *pixel* de céu limpo, *pixel* de nuvem fina e *pixel* de nuvem espessa, por exemplo);
- *Advanced Image Segmentation* (Segmentação Avançada de Imagens), onde se divide a imagem em *pixels* agrupados por características, seja para estabelecer um limite adaptável para aplicar técnicas de *thresholding*, seja para segmentar as nuvens diretamente;
- *Machine Learning* (Aprendizado de Máquina), onde se extraem características como cor, posição e textura das nuvens, que são usadas como entrada para algoritmos supervisionados de *machine learning*.

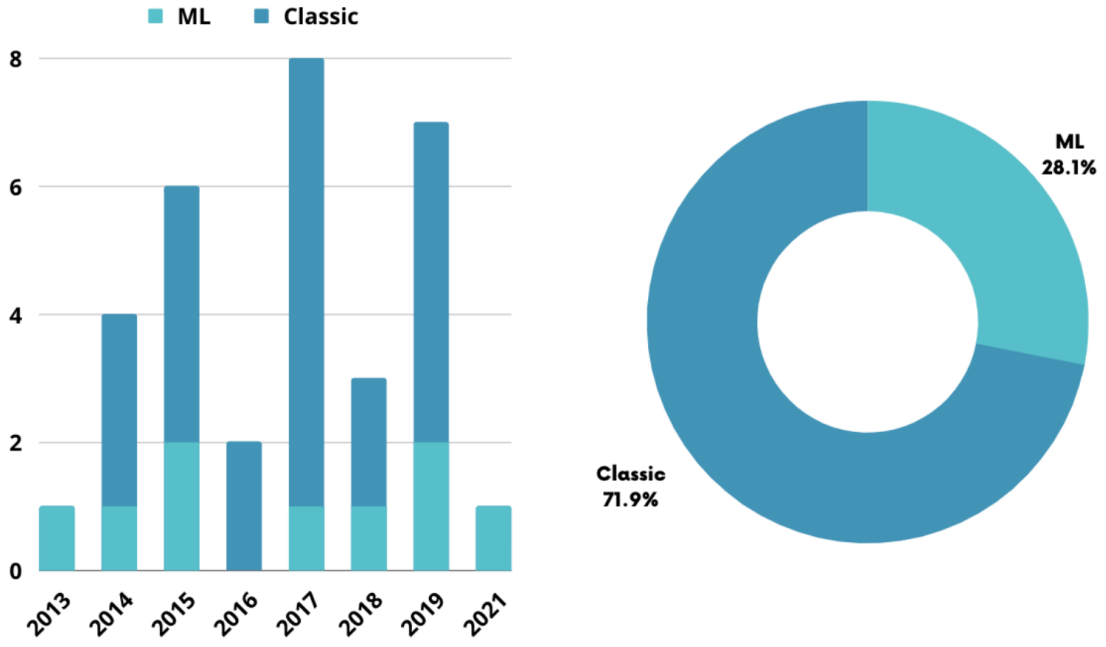
Tanto as informações trazidas por Lin, Zhang e Wang (2023) quanto por Arrais *et al.* (2022) sugerem que técnicas de *machine learning* ainda são pouco utilizadas no campo de reconhecimento de nuvens. Enquanto, segundo Lin, Zhang e Wang (2023), a técnica de segmentação de nuvens mais comum é a de *Thresholding*, Arrais *et al.* (2022) apresenta dados estatísticos, que podem ser vistos na Figura 3, demonstrando que somente 28,1% dos artigos analisados empregam *machine learning*. Além disso, observa-se que há um número significativo de artigos publicados em anos recentes, especialmente em 2017 e em 2019, onde se emprega métodos clássicos em vez de *machine learning*, o que reforça a ideia de que esta abordagem ainda é subutilizada. Por isso, no presente trabalho, buscou-se explorar o uso de *machine learning* no reconhecimento de nuvens.

2.2.2 Modelagem de movimento de nuvens

Assim como o reconhecimento de nuvens é visto como um problema de segmentação semântica, o problema da modelagem de movimento de nuvens é tratado como um problema de registro de imagens em Visão Computacional. Uma aplicação conhecida de registro de imagens é a análise de movimento de objetos (Lucas; Kanade, 1981), então é natural que se adote este tipo de técnica para rastrear o movimento de nuvens. O problema de registro de imagens consiste em encontrar o deslocamento ideal (vetor de disparidade) que torna duas imagens o mais semelhantes quanto for possível em uma região de interesse específica, medindo a diferença entre os valores de *pixel* após o deslocamento (Lucas; Kanade, 1981).

De forma semelhante com o que ocorre no reconhecimento de nuvens, o registro de imagens para modelagem de movimento de nuvens também esbarra em obstáculos

Figura 3 – Número de artigos que usaram *machine learning* x métodos clássicos



Fonte: Arrais *et al.* (2022).

relacionados as características específicas das nuvens. As nuvens movem-se de maneira complexa: mudam de velocidade em diferentes alturas, sobrepõem-se umas às outras, deformam-se, dividem-se e unem-se. Além disso, problemas de hardware podem trazer problemas de iluminação que impactam diretamente em algoritmos de modelagem de movimento de nuvens (Lin; Zhang; Wang, 2023).

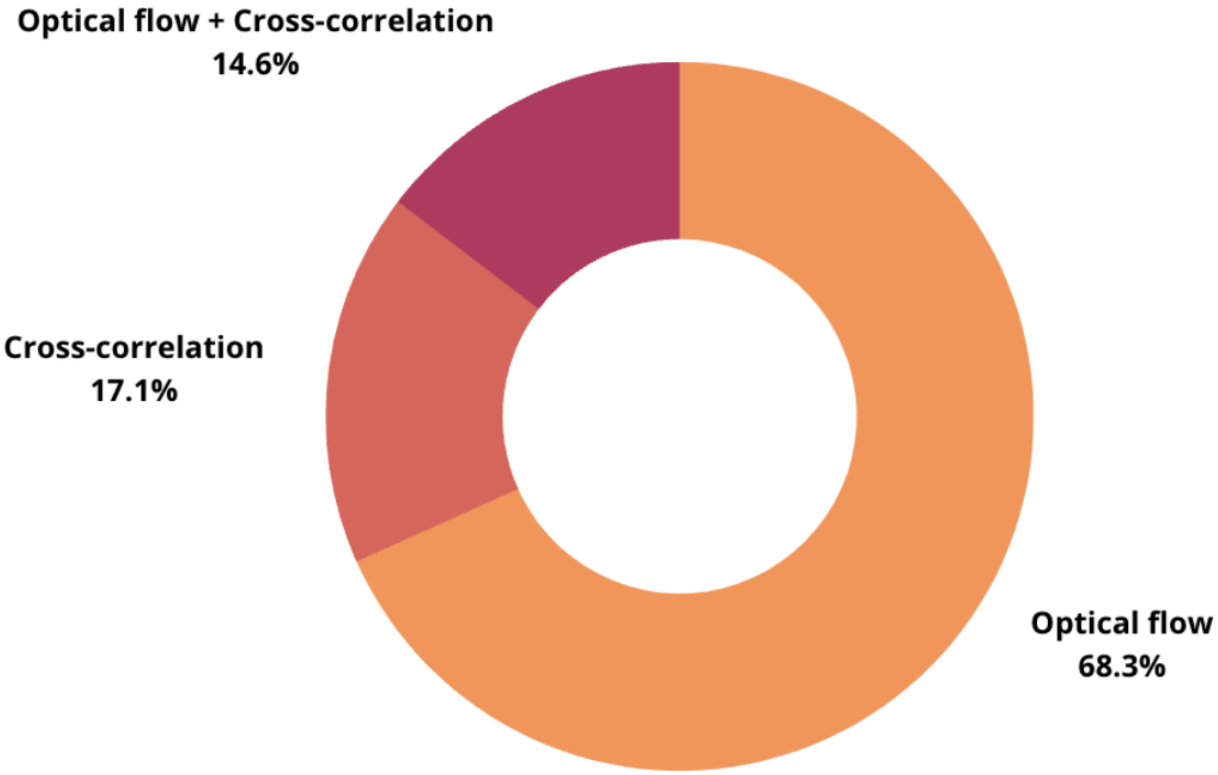
O estudo de Arrais *et al.* (2022), ao tratar dos diferentes tipos de abordagem para modelagem de movimentação de nuvens, divide os artigos entre os que usam algoritmos de fluxo óptico, os que usam *cross-correlation* (correlação cruzada) e os que usam uma combinação das duas técnicas.

Cross-correlation trata-se de uma abordagem de comparação da similaridade entre duas imagens, onde se desliza um *kernel* sobre a imagem, calculando o produto escalar a cada passo. Alguns dos trabalhos que empregam correlação cruzada usam um método baseado em blocos, onde busca-se minimizar a diferença quadrática total, a diferença quadrática absoluta, ou ambas, enquanto incrementa-se a correlação cruzada normalizada (Arrais *et al.*, 2022).

Já os métodos de fluxo ótico baseiam-se na suposição de que, em uma sequência temporal de imagens, os *pixels* em uma vizinhança local apenas mudam de posição, mas não de intensidade. Com base nessa suposição, eles procuram calcular um vetor de deslocamento para cada *pixel* em uma imagem, de modo que, quando esse vetor é aplicado aos *pixels*, a imagem resultante corresponda à próxima imagem da sequência (Lin; Zhang; Wang, 2023).

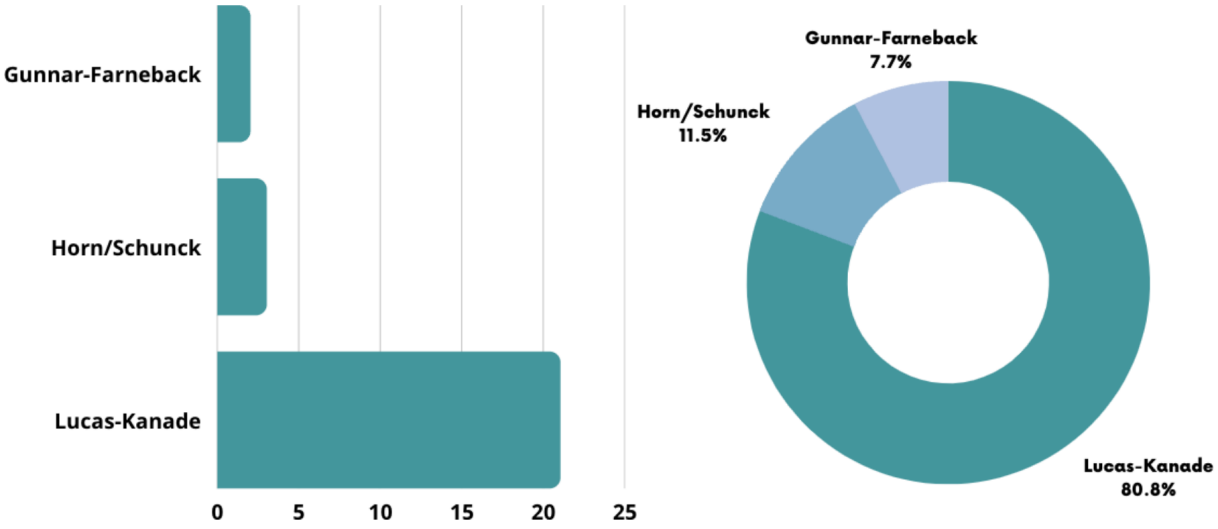
A Figura 4 mostra que, dentre os trabalhos analisados por Arrais *et al.* (2022), apenas 17,1% usaram correlação cruzada, enquanto 14,6% combinaram correlação cruzada com fluxo óptico. A grande maioria dos artigos (68,3%), empregou técnicas de fluxo óptico para detectar movimento de nuvens. Quanto aos o métodos de fluxo ótico utilizados, apresentados na Figura 5, o mais comum foi o de Lucas-Kanade, que aparece em 80,8% dos trabalhos, seguido pelo método de Horn-Schunck, utilizado em 11,5% dos artigos. O algoritmo de fluxo óptico de Gunnar-Farneback foi empregado somente em 7,7% dos trabalhos.

Figura 4 – Número de artigos que usaram fluxo óptico x *cross-correlation*



Fonte: Arrais *et al.* (2022).

Figura 5 – Métodos empregados em artigos que usam fluxo óptico



Fonte: Arrais *et al.* (2022).

3 METODOLOGIA

Esta seção traz informações materiais e métodos utilizados na presente pesquisa. Inicialmente, será descrito o conjunto de dados de 1000 imagens usado no treinamento da rede neural, onde serão especificados local, câmeras e processo de coleta de imagens, onde se detalhará o processo de anotação destas imagens e onde serão apresentadas a classificação de nuvens escolhida e a distribuição de classes. A seguir, a implementação da solução para *nowcasting* será descrita, onde serão apresentados os recursos utilizados no desenvolvimento, o processo de treinamento da rede neural para segmentação semântica, a avaliação e correção de algumas dificuldades encontradas no processo de classificação, o rastreamento do movimento de nuvens, os vídeos gerados após o processo de classificação e rastreio de movimento das nuvens e, por fim, o funcionamento e argumentos do programa em Python aqui apresentado.

3.1 CONJUNTO DE DADOS

A rede neural utilizada no presente trabalho foi treinada utilizando o *dataset* Clouds-1000[1], que é um conjunto de dados de 1000 imagens de nuvens do céu no horizonte, desenvolvido pelo Laboratório de Processamento de Imagens e Computação Gráfica (Lapix)[2] da UFSC (Universidade Federal de Santa Catarina).

As imagens foram capturadas no período entre 21 de março de 2021 e 21 de junho de 2021, utilizando um equipamento de baixo custo desenvolvido pelos pesquisadores. Esse equipamento consiste em duas câmeras Raspberry Pi modelo V1.3, executando um sistema operacional personalizado chamado Nimbus Gazer, posicionadas para capturar o céu e as nuvens que se aproximam. Ambos os equipamentos são encapsulados em uma proteção de alumínio para evitar danos causados por mudanças climáticas, e na frente há uma proteção de acrílico transparente para permitir a captura das imagens.

Cada câmera tem um campo de visão de 65 graus e estão apontadas para direções opostas, resultando em uma visão do céu equivalente a 130 graus. A Figura 6 mostra o equipamento. Durante o processo de captura, o equipamento estava conectado ao laboratório de pesquisa por meio de uma conexão *ethernet*.

Para a captura das imagens, foi escolhida uma área específica, com boa visibilidade do céu, no campus da Universidade Federal de Santa Catarina. A localização exata onde as câmeras foram instaladas pode ser vista na Figura 7. As coordenadas geográficas do local são 27°36'28.1"S 48°30'48.5"W.

Com base na análise dos ventos, estações do ano e tipos de nuvens, a equipe de pesquisadores decidiu direcionar as câmeras para o norte e sul. Essas direções foram escolhidas por serem onde predominam os ventos e, consequentemente, permitirem uma

[1] https://data.mendeley.com/datasets/4pw8vfsnpx/1
[2] https://lapix.ufsc.br/

Figura 6 – Equipamento de baixo custo desenvolvido pelo grupo de pesquisa. Na figura, o equipamento está instalado no Laboratório de Fotovoltaica da Universidade Federal de Santa Catarina

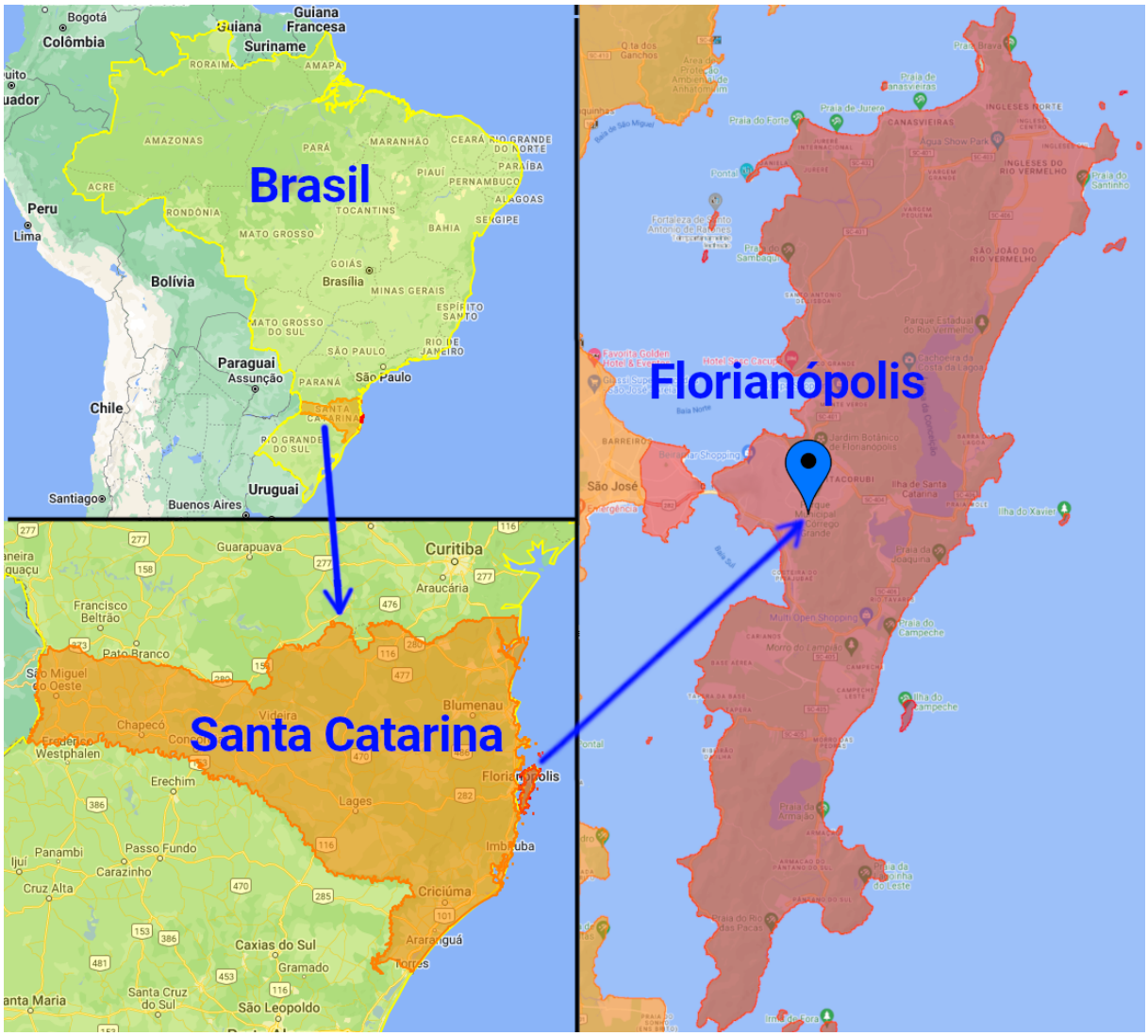
Fonte: Elaborado pelo LAPIX (2022).

visualização mais clara da aproximação das nuvens. A Figura 8 traz exemplos de imagens capturadas nas direções norte e sul.

3.1.1 Anotação

Após a captura das imagens, especialistas realizaram anotações identificando a localização e o tipo de cada nuvem no horizonte. Essas informações foram disponibilizadas

Figura 7 – Local da cidade de Florianópolis onde as imagens foram capturadas



Fonte: Elaborado pelo autor (2023).

em um formato compreensível para algoritmos de *machine learning*. A equipe responsável pelas anotações era composta por 3 analistas de dados – meteorologistas juniores encarregados de selecionar e anotar as imagens das nuvens – e 2 meteorologistas seniores, responsáveis por revisar as anotações e corrigir possíveis erros nas imagens com problemas.

As anotações foram realizadas manualmente usando a ferramenta Supervisely[3], um *software* de gerenciamento e anotação de imagens. O Supervisely permite a anotação das imagens por meio de uma interface *web*, semelhante a um editor de imagens, como outras ferramentas disponíveis no mercado. As nuvens de cada imagem foram anotadas utilizando a ferramenta de polígono do Supervisely e classificadas de acordo com as classes que serão descritas na próxima seção. Exemplos de imagens anotadas podem ser vistos na Figura 9.

[3] https://supervisely.com/

Figura 8 – Exemplos de imagens capturadas por câmeras apontando para o horizonte, voltadas para as direções norte (esquerda) e sul (direita).

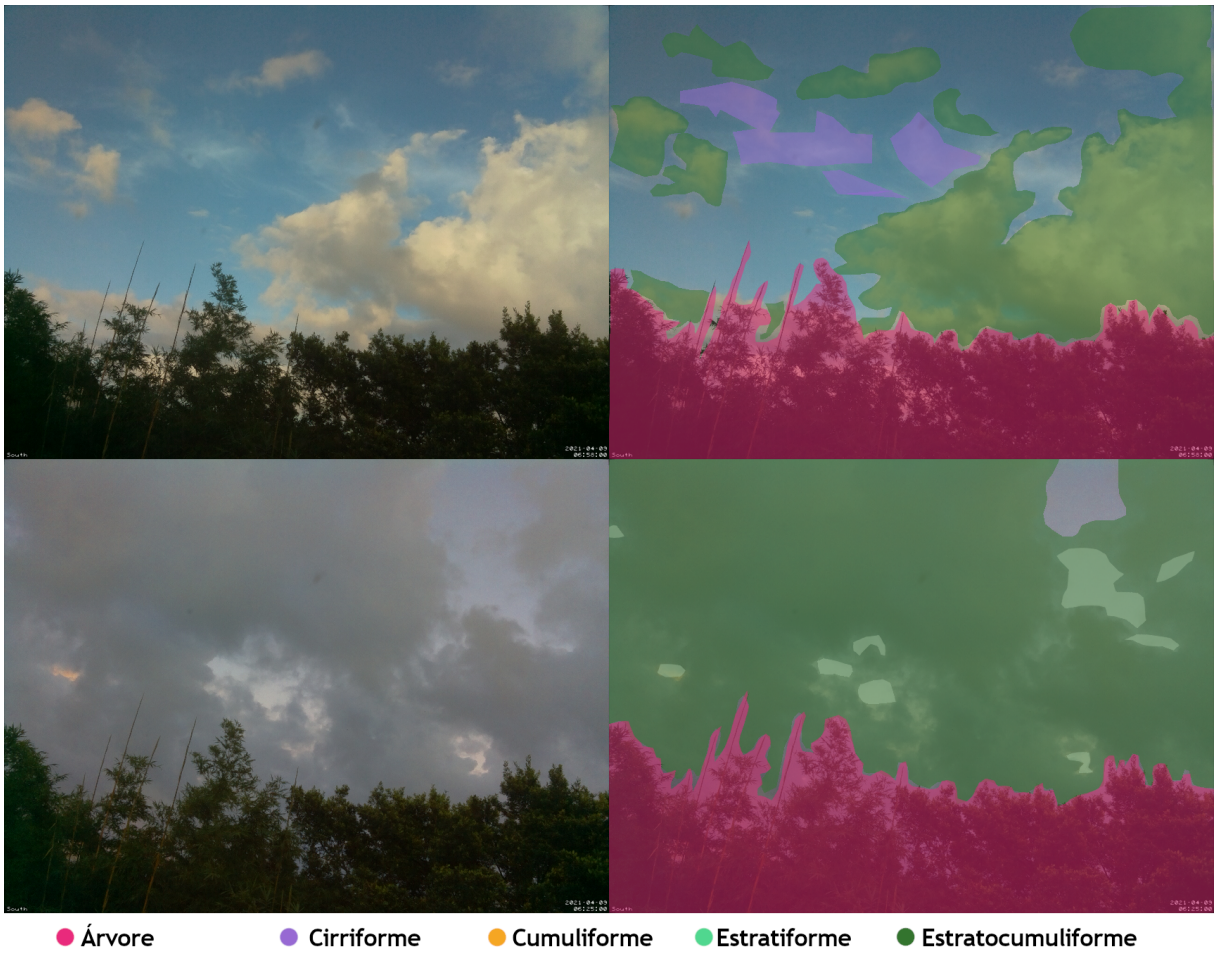
Fonte: Elaborado pelo autor (2023).

O processo de anotação com o Supervisely demanda bastante tempo e está sujeito a erros, principalmente devido ao formato das nuvens. Mesmo com o trabalho de especialistas, a classificação leva tempo, pois características como o tamanho e densidade das nuvens podem não ser claras, gerando dúvidas sobre o formato e a classificação de cada nuvem. A dificuldade em lidar com o formato das nuvens também atrasa o trabalho de revisão dos especialistas, fazendo com que a etapa de revisão de dados seja realizada por amostragem, ou seja, nem todas as imagens do conjunto de dados foram validadas. Além disso, o processo mecânico de anotação de imagens requer que o anotador circule manualmente as nuvens utilizando a ferramenta de polígono, o que consome bastante tempo, especialmente quando as nuvens possuem formatos complexos, apresentam buracos ou estão sobrepostas.

A seguir, segue resumo do *workflow* do processo de captura e anotação de imagens usado na criação do *dataset* Clouds-1000. A Figura 10 traz um diagrama apresentando o *workflow* completo.

a) Captura de imagens: captura de imagens das nuvens, usando câmeras Raspberry

Figura 9 – Exemplos de anotações do Supervisely. À esquerda, imagens originais e, à direita, imagens anotadas.



Fonte: Elaborado pelo autor (2023).

Pi e o sistema operacional Nimbus Gazer;

b) Seleção de dados: seleção manual das imagens com boa visibilidade pelos analistas de dados;

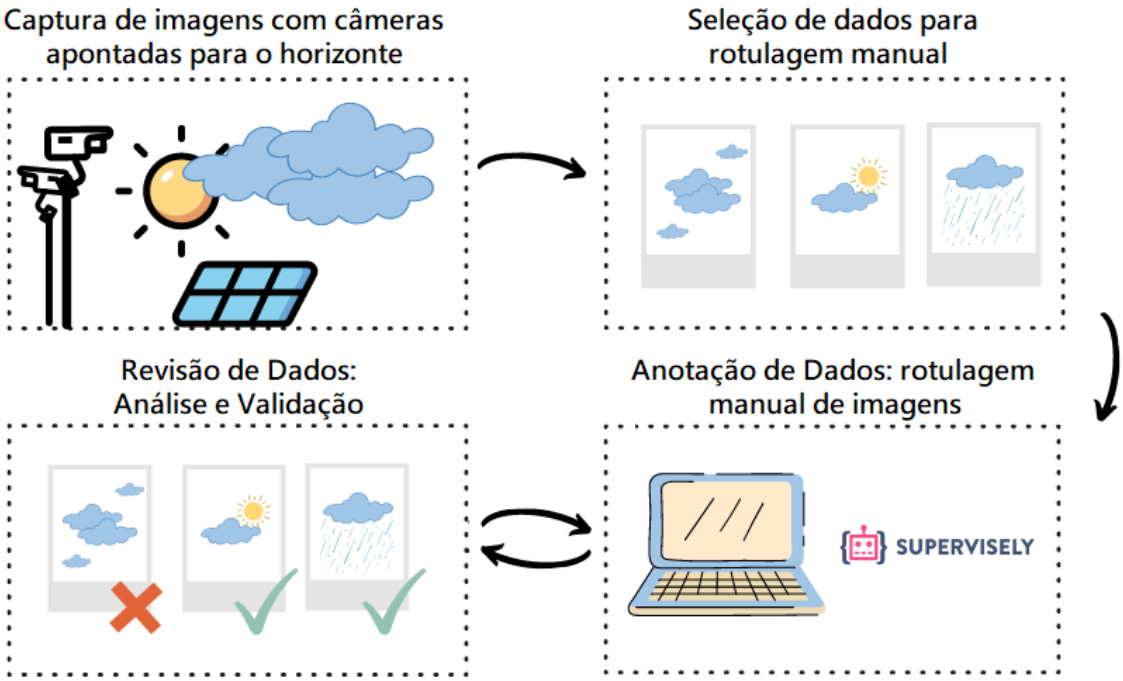
c) Anotação de dados: anotação de imagens feita pelos analistas de dados via Supervisely;

d) Revisão de dados: revisão das anotações por meteorologistas, retornando ao passo *c* quando identificadas imagens com problemas de anotação e que demandam correção.

3.1.2 Classificação

Os tipos de nuvem empregados para classificação de imagens do Clouds-1000 correspondem às cinco famílias de nuvens propostas no trabalho de Barrett e Grant (1976).

Figura 10 – *Workflow* do processo de captura e anotação de imagens.



Fonte: Elaborado pelo LAPIX (2022).

Estas famílias, por sua vez, são agrupamentos dos gêneros de nuvem da categorização da WMO. De acordo com o *WMO Cloud Atlas*[4], a classificação de nuvens da WMO baseia-se na aparência da nuvem, o que inclui suas dimensões, formato, estrutura, textura, luminância e cor. As famílias de Barrett e Grant (1976), por sua vez, agrupam as categorias da WMO usando a textura das nuvens como critério. A lista de famílias e dos gêneros que cada uma delas engloba podem ser vistos na Quadro 1.

Quadro 1 – Famílias de nuvens e categorias englobadas por cada uma delas.

Família	**Categorias**
Cumulonimbiforme	*Cumulonimbus* e *Cumulonimbus* com *Cirrus*
Cumuliforme	*Cumulus humilis*, *Cumulus mediocris*, *Cumulus congestus* e *Altocumulus*
Estratiforme	*Stratus*, Estratiformes em camadas (inclusive *Nimbostratus*), *Altostratus*
Estratocumuliforme	*Stratocumulus*
Cirriforme	*Cirrus fibratus*, *Cirrus spissatus*, *Cirrostratus*, *Cirrocumulus* e rastro de condensação

Fonte: Elaborada pelo autor (2023).

Além das classes correspondentes às famílias de nuvem, também incluiu-se nas

[4] https://cloudatlas.wmo.int/en/appearance-of-clouds.html

anotações do *dataset* a classe *Árvore*. Foram anotados com esta classe não só as árvores, mas também outros elementos que obstruem a visão do céu, como prédios, postes de luz, etc.

Devido ao fato de a seleção das nuvens presentes nas imagens estar restrita às condições climáticas da região e ao período de captura das imagens, há certo desequilíbrio nos dados, com algumas famílias de nuvens sendo mais comuns do que outras. Por exemplo, nuvens do tipo estratocumuliforme aparecem em 81,53% das imagens: quantidade muito maior do que a dos demais tipos. Por outro lado, nuvens do tipo cumulonimbiforme, típicas de climas secos, não apareceram em nenhuma das imagens. No Quadro 2 temos a quantidade de imagens em que cada classe aparece no *dataset*, incluindo a classe Árvore.

Quadro 2 – Quantidade de nuvens de cada família presentes no *dataset*.

Classe	**Quantidade de imagens**	**Quantidade de imagens (%)**
Árvore	989	99,3
Estratocumuliforme	812	81,53
Estratiforme	271	27,21
Cirriforme	285	28,61
Cumuliforme	90	9,04
Cumulonimbiforme	0	0,00

Fonte: Elaborada pelo autor (2023).

Devido ao fato de não existirem nuvens do tipo Cumulonimbiforme no *dataset*, na prática, existem apenas 5 tipos de classes de pixel presentes nas anotações: a classe *Árvore* e as classes correspondentes às outras 4 famílias de nuvem da classificação de Barrett e Grant (1976).

3.2 DESENVOLVIMENTO DA SOLUÇÃO

O desenvolvimento do *software* de classificação e rastreamento de movimento de nuvens envolveu o treinamento de um segmentador semântico, a implementação de uma rotina de correção de erros de segmentação conhecidos, a aplicação do algoritmo de fluxo óptico sobre o resultado da segmentação corrigido e a geração de vídeos que apresentam a classificação e a direção do movimento das nuvens. Essas etapas combinadas resultaram em um *script* escrito em Python, o *cloudseg.py*. O desenvolvimento do *script*, que unifica as funcionalidades mencionadas acima, será detalhado a seguir.

3.2.1 Recursos

O *script* apresentado neste trabalho foi implementado na versão 3.10.11 da linguagem Python, usando a IDE Spyder[5] para o desenvolvimento. Ele se encontra no Apêndice B

[5] https://www.spyder-ide.org/

do presente trabalho, e está disponível em repositório do GitHub [6]. Abaixo, são listadas as bibliotecas empregadas no desenvolvimento da solução:

- PaddleSeg[7] versão 2.8.0: *framework* de segmentação de imagens, baseado na plataforma de *deep learning* PaddlePaddle[8], onde se encontra a implementação do PP-LiteSeg: modelo usado no presente trabalho para classificação de nuvens.
- OpenCV[9] versão 4.5.5.64: biblioteca de Visão Computacional, usada para ler imagens, manipular cores, calcular componentes conexas e fluxo óptico de imagens e para gerar o vídeo de saída do programa.
- FFmpeg[10] versão 4.3.1: grava *streams* de áudio e vídeo, necessária para geração do vídeo via OpenCV.
- NumPy[11] versão 1.24.3: biblioteca para computação científica com funcionalidades para cálculos envolvendo matrizes, o que é necessário para a manipulação de imagens do OpenCV.

3.2.2 Segmentador Semântico

Para a identificação e classificação das nuvens nas imagens, foi escolhido o modelo de rede neural PP-LiteSeg. Essa decisão foi baseada nos resultados no estado da arte alcançados pelo modelo, conforme descrito em seu artigo original, onde verifica-se que o PP-LiteSeg apresentou melhor equilíbrio entre velocidade de inferência e intersecção sobre união média em testes realizados com vários modelos diferentes de segmentação semântica, nos *datasets* CamVid e Cityscapes (Peng *et al.*, 2022).

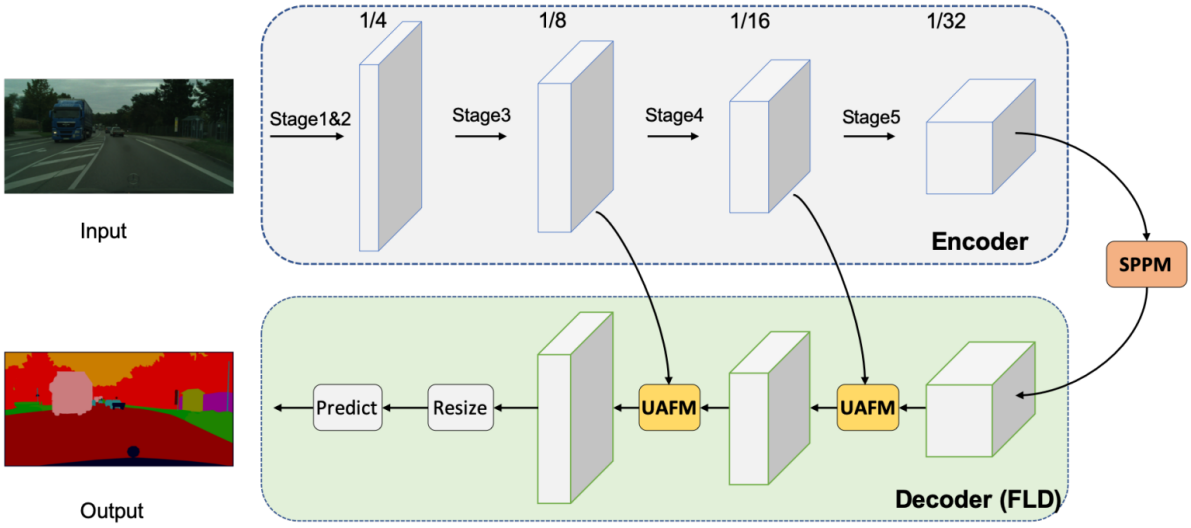
Conforme explicado na subseção anterior, o treinamento do modelo foi realizado utilizando o PaddleSeg. Optou-se por esse *framework* por ele ter sido desenvolvido pelos mesmos criadores do PP-LiteSeg, e ser o único para o qual há implementação disponível deste modelo.

3.2.2.1 Arquitetura do modelo

O PP-LiteSeg é um modelo de rede neural projetado para segmentação semântica que utiliza arquitetura *encoder-decoder*, como evidenciado pela Figura 11. O modelo possui três módulos essenciais: *encoder*, agregação e *decoder* (Peng *et al.*, 2022).

[6] https://github.com/GilbertoRicci/CloudSeg
[7] https://github.com/PaddlePaddle/PaddleSeg
[8] https://github.com/PaddlePaddle/
[9] https://opencv.org
[10] https://www.ffmpeg.org
[11] https://numpy.org

Figura 11 – Arquitetura do modelo de segmentação semântica PP-LiteSeg



Fonte: Peng *et al.* (2022).

No estágio do *encoder*, faz-se a extração de *features* utilizando-se um modelo de rede neural leve como *backbone*. O artigo original usa o modelo STDCNet para o *encoder*, devido a sua performance, onde há testes utilizando STDC1 e STDC2 (Peng *et al.*, 2022).

A agregação de contexto, por sua vez, é realizada por meio do Simple Pyramid Pooling Module (SPPM), um componente desenvolvido com base no *framework* PPM ajustado para permitir execução em tempo real. Este módulo aplica à saída do *encoder* três operações de *global-average pooling*, com *bin sizes* diferentes, que são somadas e integradas à entrada do *decoder* (Peng *et al.*, 2022).

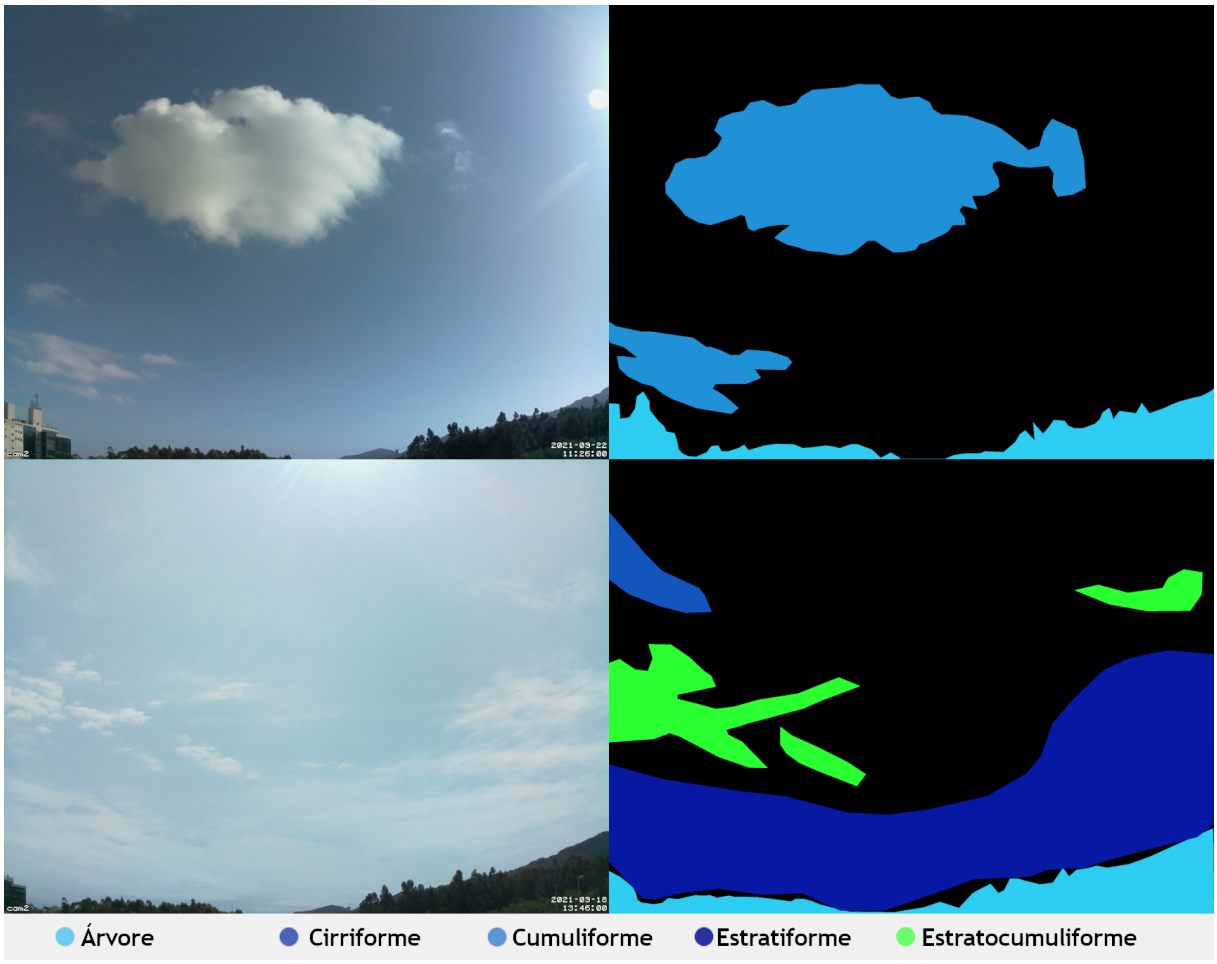
Por ultimo, o PP-LiteSeg usa a arquitetura Flexible and Lightweight Decoder (FLD) para o *decoder*. Ao contrário dos *decoders* convencionais para segmentação semântica, que expandem a dimensionalidade espacial das *features* enquanto mantém o número de canais constante, o FLD reduz gradualmente o número de canais durante o *upsample* (Peng *et al.*, 2022).

Para garantir que o *upsample* com redução de canais seja feito sem comprometer a acurácia do modelo, o PP-LiteSeg emprega módulos de arquitetura Unified Attention Fusion Module (UAFM) na fusão de *features* de alto e baixo nível. O UAFM usa mecanismos de atenção para gerar pesos, aplicados às *features* de cada camada do *encoder* e do *decoder*, antes de sua soma. O artigo propões dois tipos de módulos UAFM: espacial e de canais, destinados a explorar relações inter-espaciais e inter-canais, respectivamente (Peng *et al.*, 2022).

3.2.2.2 Treinamento

O conjunto de dados utilizado para treinar a rede neural foi o Clouds-1000, mencionado anteriormente. Para que os dados pudessem ser utilizados no treinamento do modelo via PaddleSeg, foi utilizada a ferramenta *Export as masks* do Supervisely. Esta ferramenta exporta os dados de imagens e anotações salvos no servidor do Supervisely e os disponibiliza para download. Usou-se este método para ter acesso ao *dataset* devido ao fato dele exportar as anotações das imagens no formato de imagens PNG 8-bits, onde cada cor representa uma classe de anotação, sendo que as áreas totalmente pretas correspondem a áreas não anotadas. Exemplos de imagens neste formato, comparadas com as imagens originais, podem ser vistas na Figura 12. Este formato é conveniente pois pode ser convertido facilmente em um *array* de números inteiros pela biblioteca NumPy. Ao serem convertidas para esse formato, que é compatível com o PaddleSeg, as anotações podem ser usadas pela ferramenta para o treinamento da rede neural.

Figura 12 – Imagens capturadas (à esquerda) comparadas com as anotações importadas do Supervisely no formato PNG 8 bits (à direita).



Fonte: Elaborado pelo autor (2023).

A divisão do *dataset* em conjuntos de treinamento, validação e teste foi feita aleatoriamente pelo próprio PaddleSeg, nas proporções de 80%, 10% e 10% respectivamente.

O treinamento do modelo foi executado em 80.000 iterações, com *batch size* igual a 4, utilizando-se o modelo STDC2 como *backbone* do *encoder*. Para acelerar o treinamento, as imagens tiveram suas dimensões reduzidas pela metade antes de alimentar a rede, resultando em uma resolução de 1296x962. Além disso, foram aplicadas técnicas de *data augmentation* para melhorar a qualidade do modelo, porém, com a cautela de usar transformações que não descaracterizam as imagens. Assim, usou-se *random horizontal flip*, mas não o vertical, já que, em imagens reais, o chão nunca está acima das nuvens. Também foram aplicadas transformações no brilho, contraste e saturação da imagem, com variação máxima de 25%. Os dados foram normalizados para o treinamento.

O treinamento do modelo foi realizado em um computador com Windows 10, equipado com um processador AMD Ryzen 7 1700X 3,4GHz, 16GB de RAM e uma placa de vídeo NVIDIA GeForce GTX 1070. Devido às limitações de configuração do dispositivo, o tempo total de treinamento foi de aproximadamente 3 dias.

A função de perda utilizada foi a *MultiClass Focal Loss* e *Dice Loss* (*F1 Score*), com o mesmo peso para ambas. O modelo foi treinado com uma taxa de aprendizado de 10^{-4}, utilizando *Polynomial Decay* com um fator de 0,9, e otimizador AdamW, com betas de 0,9 e 0,999 e taxa de decaimento de 0,01. O ajuste dos valores dos hiperparâmetros foi feito empiricamente, onde buscou-se balancear tempo de treino com a eficácia do modelo, dadas as limitações da máquina usada para o treinamento.

Após o treinamento, os pesos do modelo com a maior acurácia alcançada foram exportados utilizando ferramenta disponível no PaddleSeg. Esses pesos são salvos em um formato compatível com a ferramenta de inferência do *framework* e são responsáveis pela segmentação semântica das nuvens na solução desenvolvida.

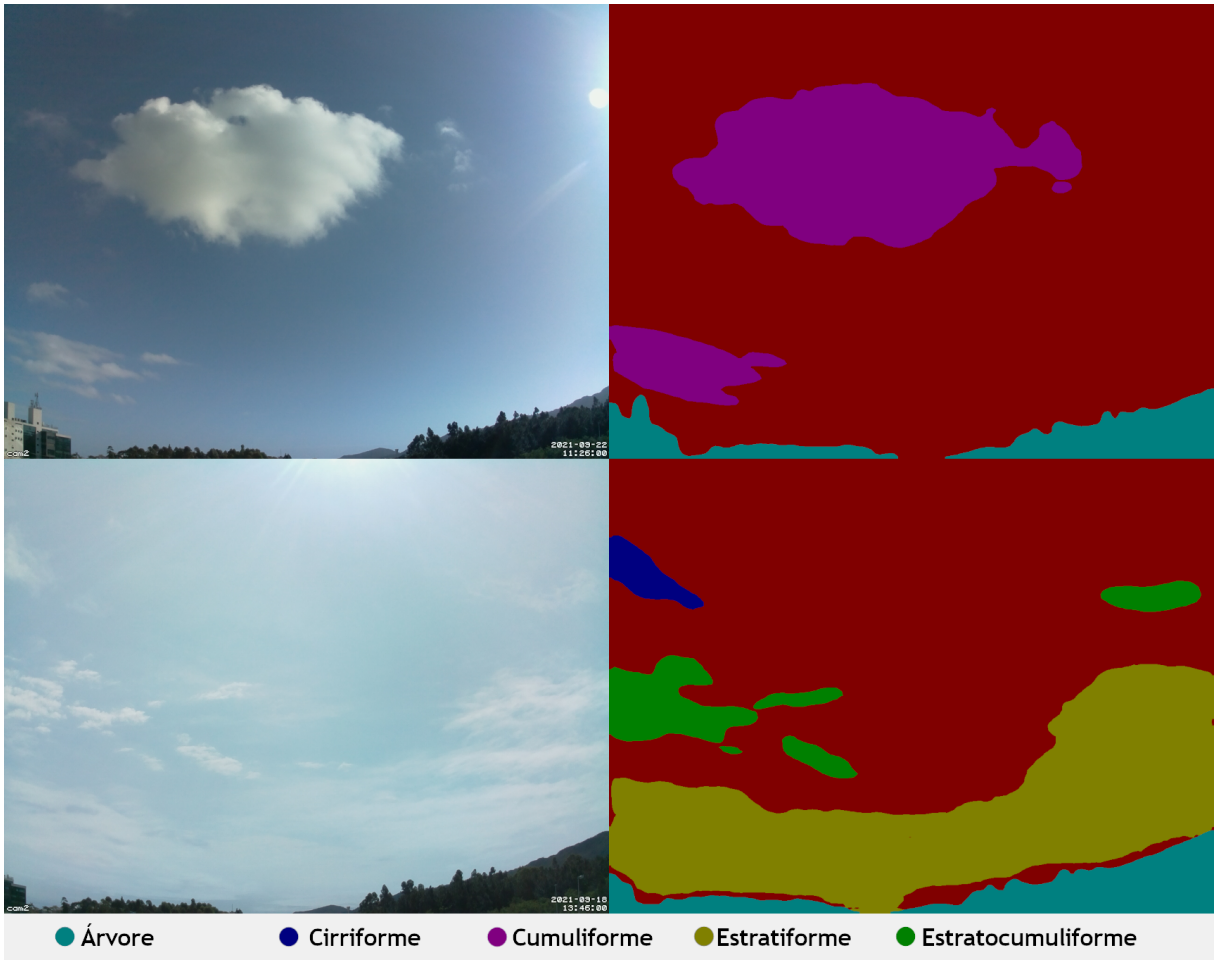
Ao ser executado sobre um conjunto de imagens, o segmentador semântico gera imagens de nuvens classificadas e segmentadas, imagens estas que são salvas em um diretório do disco e são usadas em etapas posteriores do *script* para a geração do vídeo. A Figura 13 mostra exemplos das imagens geradas na segmentação semântica.

3.2.3 Correção de problemas de classificação

Durante a análise qualitativa das classificações obtidas pelo modelo treinado, foram observados alguns problemas. Em alguns casos, fragmentos de nuvens eram erroneamente classificados como um tipo diferente do restante da nuvem, como se houvesse uma nuvem dentro da outra, conforme exemplificado na Figura 14. Além disso, em algumas imagens, uma nuvem era classificada como um determinado tipo, mas na imagem subsequente, era classificada como outro tipo, como se a classe da nuvem pudesse mudar de minuto a minuto, como pode ser visto na Figura 15.

Problemas deste tipo inviabilizam o uso de algoritmos de fluxo óptico e, conse-

Figura 13 – Imagens antes (à esquerda) e depois (à direita) da segmentação semântica.



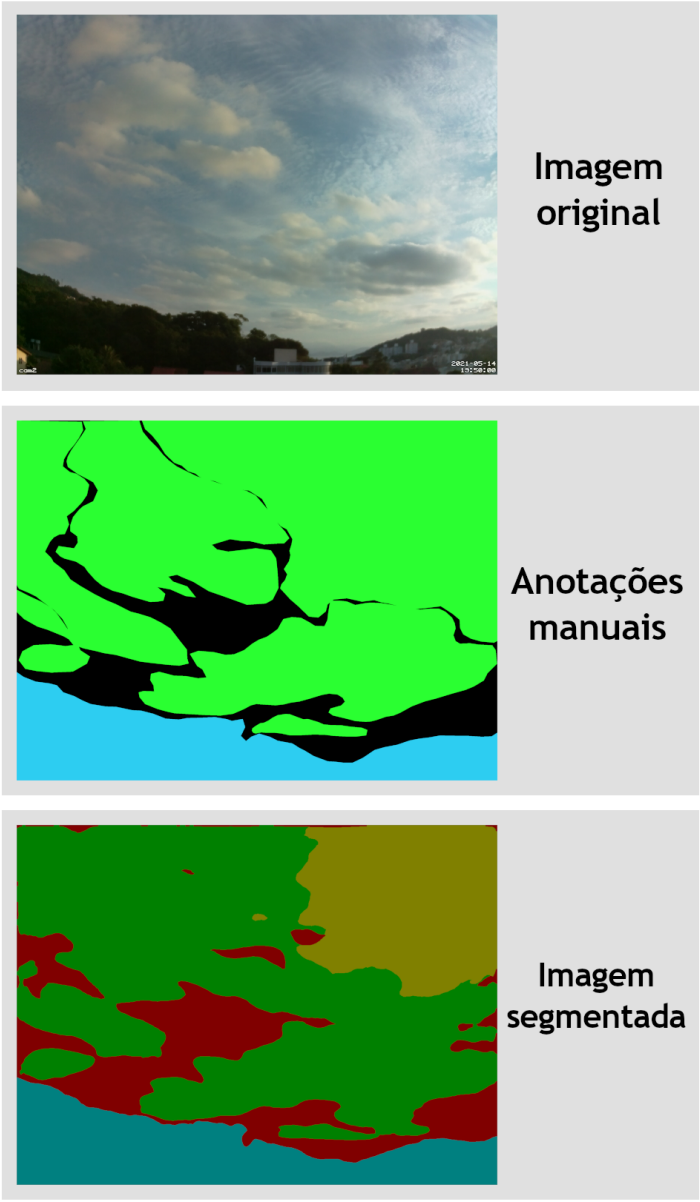
Fonte: Elaborado pelo autor (2023).

quentemente, o rastreamento do movimento das nuvens. Isso ocorre porque este tipo de algoritmo parte da premissa de que a intensidade dos *pixels* permanece constante ao longo do tempo. As confusões nas classificações das nuvens em diferentes *frames* resultam em uma mudança na intensidade dos *pixels* de um mesmo objeto na passagem de um *frames* para o outro. Esta mudança de intensidade faz os métodos de fluxo óptico perderem o rastro dos *pixels* do objeto.

Para lidar com esses problemas, foi necessário realizar um pós-processamento das imagens classificadas antes de rastrear o movimento das nuvens. Esse processo consiste em garantir que as nuvens fragmentadas em várias classes sejam consideradas como uma única classe, alterando sua classificação para a classe predominante da nuvem. O cálculo da classe predominante deve levar em conta a classificação da nuvem nos *frames* anteriores, de modo a solucionar-se também o problema de mudança de classe das nuvens ao longo do tempo.

Esse pós-processamento é realizado por meio de rotinas do *script* em Python

Figura 14 – Exemplo do problema de fragmentação de nuvens. Na imagem segmentada, a nuvem foi fragmentada erroneamente em duas classes, resultando em classificação diferente das anotações manuais.

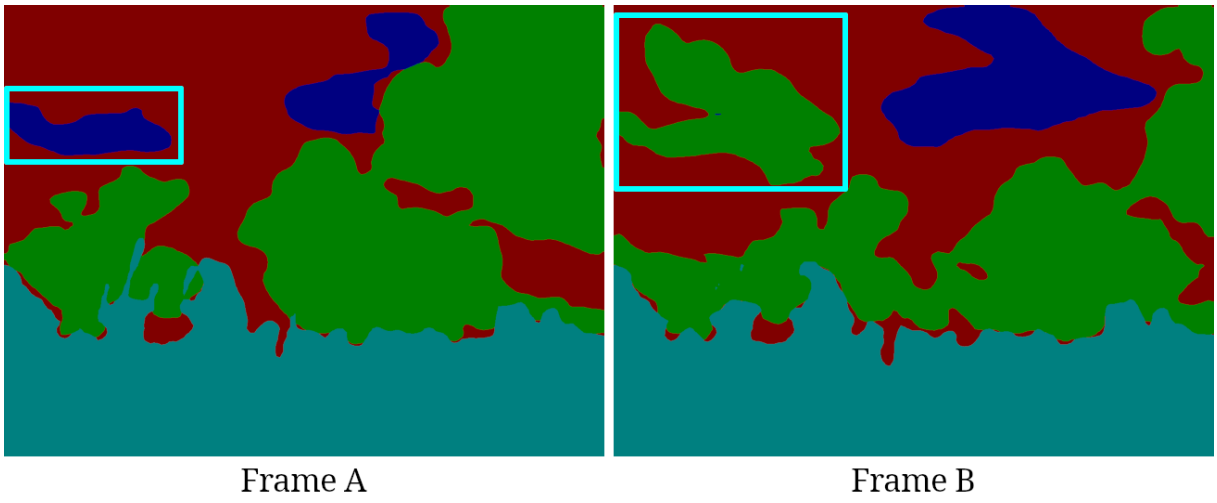


Fonte: Elaborado pelo autor (2023).

desenvolvido. Seu primeiro passo é identificar cada uma das nuvens. Para isso, utiliza-se o método que encontra as componentes conexas da imagem. Neste contexto, considera-se que cada componente conexa representa uma nuvem.

Após encontrar as componentes conexas, que correspondem a cada nuvem individualmente, é realizado o cálculo da classe predominante. Esse cálculo leva em consideração tanto a imagem em processo de pós-processamento quanto as imagens dos dois *frames* anteriores.

Figura 15 – Exemplo de imagens segmentadas com problema de mudança de classe. Na figura, a nuvem destacada muda de classe na passagem do *frame* A ao *frame* B.



Fonte: Elaborado pelo autor (2023).

O cálculo da classe predominante consiste em uma contagem de *pixels*: verifica-se quantos *pixels* de cada classe a nuvem possui, tanto no *frame* atual quanto nos dois *frames* anteriores, e elege-se a classe que aparece no maior número de *pixels* como sendo a classe predominante.

A contagem de *pixels* de uma nuvem é feita usando sua componente conexa para criar uma máscara com o formato da nuvem/componente. Aplicando esta máscara a uma imagem, é possível extrair somente os *pixels* contidos dentro do espaço onde estaria a nuvem. A seguir, através da aplicação desta máscara à imagem do *frame* atual nas imagens dos dois *frames* anteriores, obtém-se somente os *pixels* que devem ser contados. A classe de nuvem que aparecer na maioria dos *pixels* será considerada a classe predominante para aquela nuvem.

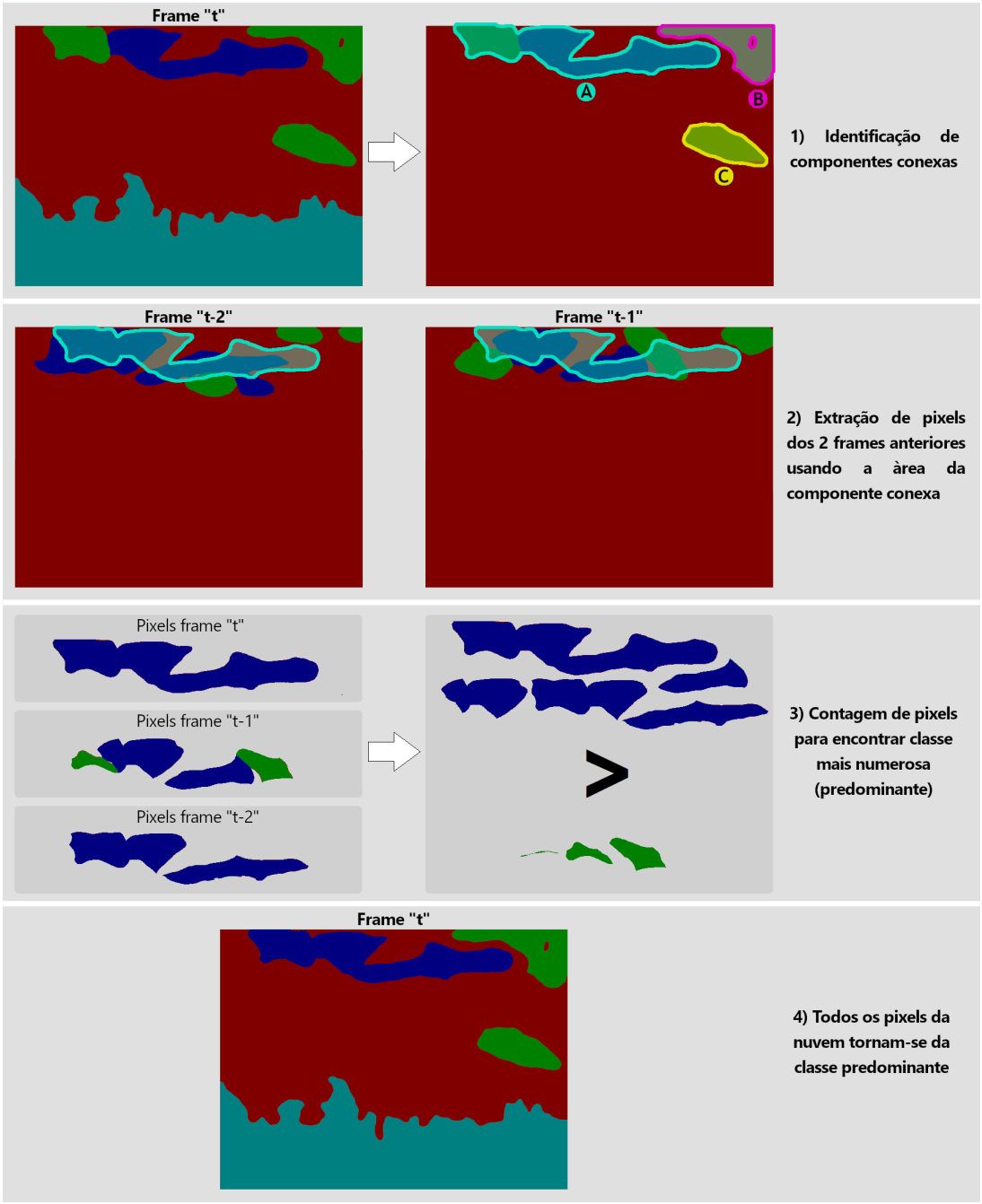
É importante destacar que, antes desse pós-processamento, a classe *Árvore* é desconsiderada, para evitar confusão entre o que é solo e o que é nuvem.

Um exemplo do funcionamento da rotina de pós-processamento pode ser visto na Figura 16.

3.2.4 Rastreamento de movimento de nuvens

O rastreamento do movimento das nuvens é realizado por meio do algoritmo de fluxo óptico de Gunnar-Farneback. Diferentemente de algoritmos de fluxo ótico esparso, como o de Lucas-Kanade, que calculam o movimento usando apenas *pixels* chave da imagem, obtidos através de algoritmos como o de Shi-Tomasi, o método de Gunnar-Farneback analisa o fluxo óptico denso da imagem, calculando o deslocamento de todos

Figura 16 – Demonstração do cálculo da classe predominante para a nuvem A.



Fonte: Elaborado pelo autor (2023).

os *pixels* entre dois *frames* consecutivos (Arrais *et al.*, 2022). Tal abordagem apresenta-se vantajosa para o rastreio de movimento de nuvens, pois, como observa Arrais *et al.* (2022), a esparsidade dos limites das nuvens, nebulosas por natureza, dificulta a identificação de bordas e cantos que possam ser usados como *pixels*-chave.

Conforme mencionado anteriormente, métodos de fluxo óptico buscam encontrar um vetor que descreva o deslocamento dos *pixels* de um *frame* para o *frame* subsequente,

partindo da premissa de que os *pixels* não mudam de intensidade. Ou seja, busca-se um vetor (v_x, v_y) onde seja verdadeira a Equação (1),

$$I(x, y, t) = I(x + v_x, y + v_y, t + 1) \tag{1}$$

onde $I(x, y, t)$ é a intensidade do *pixel* nas coordenadas x, y da imagem, e t indica a ordem temporal dos *frames* na sequência. Para encontrar o vetor, a equação Equação (1) pode ser aproximada do polinômio de Taylor de 1º grau dado por Equação (2),

$$v_x I'_x(x, y, t) + v_y I'_y(x, y, t) + I'_t(x, y, t) = 0 \tag{2}$$

deste que assuma-se que o deslocamento dos *pixels* é pequeno (Lin; Zhang; Wang, 2023).

Como a Equação (2) possui duas incógnitas, ela é indeterminada, sendo necessário incluir outras restrições ao problema para ser possível o cálculo de (v_x, v_y) (Lin; Zhang; Wang, 2023).

3.2.4.1 Fluxo óptico de Gunnar-Farneback

O método de Gunnar-Farneback estima o deslocamento de *pixels* entre dois *frames* de uma sequência de imagens aproximando cada *frame* a um polinômio quadrático, em determinada vizinhança de *pixel*. Assim, busca-se calcular um deslocamento d, capaz de transformar o polinômio correspondente ao primeiro *frame*, dado por Equação (3),

$$f_1(x) = x^T A_1 x + b_1^T x + c_1 \tag{3}$$

ao polinômio dado por Equação (4)

$$f_2(x) = f_1(x - d) = (x - d)^T A_1 (x - d) + b_1^T (x - d) + c_1 \tag{4}$$

que corresponde ao *frame* subsequente (Farnebäck, 2003).

Nas equações acima, os coeficientes A_1 e A_2 são matrizes simétricas, b_1 e b_2 são vetores e c_1 e c_2 são escalares. No artigo original, os coeficientes A_1, b_1 e c_1 são calculados através de uma versão ponderada do método dos mínimos quadrados, enquanto A_2, b_2 e c_2 podem ser obtidos da Equação (4) (Farnebäck, 2003).

Como é pouco realista assumir que um *frame* pode ser representado por completo por um polinômio, que existe uma translação que o transforma perfeitamente no polinômio do próximo *frame*, e como, na prática, o cálculo de d ponto a ponto produz muito ruído, a solução adotada no método de Gunnar-Farneback é a de encontrar um valor de deslocamento que se aplique à vizinhança de um ponto com um erro mínimo (Farnebäck, 2003).

3.2.4.2 Vetor predominante de deslocamento

Como o objeto de interesse do presente trabalho é o deslocamento das nuvens e não de *pixels* individuais, os vetores de deslocamento dos *pixels* da imagem, encontrados pelo algoritmo de fluxo óptico, são usado para obter um vetor de deslocamento predominante para cada nuvem de uma sequência de imagens do céu. Este vetor predominante indicará a magnitude e a direção do movimento da nuvem.

O cálculo do vetor predominante utiliza como base o fluxo óptico denso entre as imagens geradas pelo segmentador semântico, onde cores de pixel diferentes representam classes diferentes de nuvem. Para obtê-lo, utiliza-se implementação do algoritmo de Gunnar-Farneback da biblioteca OpenCV, que recebe como entrada duas imagens consecutivas da sequência de imagens segmentadas, retornando como saída as componentes horizontal e vertical dos vetores de deslocamento dos *pixels* da primeira imagem para a imagem subsequente. Através da soma vetorial das componentes dos vetores dos *pixels* de uma nuvem de interesse, dividida pelo número de *pixels* da nuvem, calcula-se o vetor predominante. Para identificar quais *pixels* correspondem a cada nuvem, utilizam-se as componentes conexas obtidas para o cálculo de correção de problemas de classificação, mencionado na subseção anterior.

3.2.5 Geração de vídeo

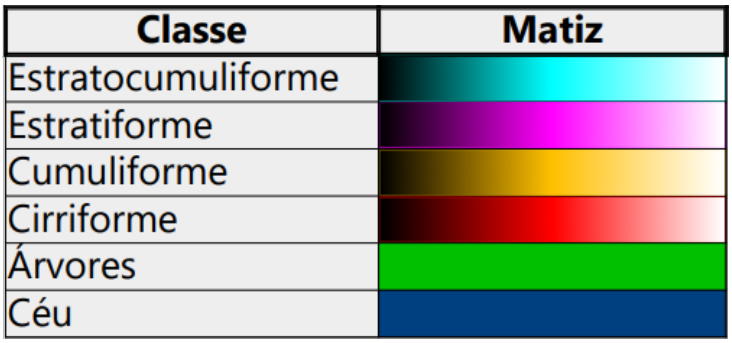
A última etapa de execução do *script cloudseg.py* é responsável pela geração do vídeo que apresenta o resultado da classificação e rastreio de movimento de nuvens. A partir das informações sobre classe de nuvem e vetor de movimento de uma sequência de imagens do céu, informações estas obtidas em etapas anteriores, a rotina de geração de vídeo cria imagens correspondes às imagens mencionadas, onde as nuvens aparecem segmentadas, classificadas e com indicação de seu movimento. Estas imagens são usadas para compor um vídeo, que será a saída do *script*.

No processo de representação da classificação e movimentação das nuvens nos vídeos gerados pelo *script*, um sistema de cores é adotado para facilitar a identificação e compreensão das características das nuvens, onde as nuvens são coloridas de acordo com sua classe e com o vetor predominante de seu deslocamento.

Neste sistema, a matiz da cor corresponde à classe da nuvem, de modo que há uma matiz de cor correspondente a cada uma das classes de pixel existentes. O céu e as árvores também são coloridos, de modo a distinguir-se uns dos outros e das nuvens. Na Figura 17, estão listadas as matizes de cor atribuídas a cada classe de *pixel*.

Por sua vez, a luminosidade da cor de uma nuvem representa a magnitude de seu vetor de deslocamento. O valor da luminosidade é calculado a partir da magnitude em *pixels* do vetor predominante da nuvem, valor este que é normalizado via algoritmo *min-max* entre zero e a magnitude máxima dentre todos os vetores predominantes encontrados

Figura 17 – Matizes de cor usadas na representação das diferentes classes.



Fonte: Elaborado pelo autor (2023).

em todas as imagens. A normalização garante que uma determinada luminosidade de cor corresponde à mesma magnitude em qualquer *frame* do vídeo. As intensidades de luminosidade podem variar de 0,3 (magnitude máxima) a 1,0 (magnitude mínima), de modo que uma nuvem escura estaria se movendo mais rápido do que uma nuvem clara, e uma nuvem totalmente branca estaria totalmente parada.

Assim, de acordo com o sistema de cores empregado, uma nuvem ciano escura, por exemplo, seria identificada como uma estratocumuliforme em movimento rápido. Já uma nuvem amarelo claro seria uma cumuliforme deslocando-se lentamente.

Setas desenhadas na nuvem, que apontam para a direção seu vetor predominante de deslocamento, indicam para onde a nuvem está se movimentando.

A Figura 18 traz um exemplo com imagens de nuvens geradas pelo *cloudseg.py* a partir de imagens de nuvens segmentadas criadas artificialmente. No exemplo, é possível visualizar a variação de cores das nuvens de acordo com sua classificação e deslocamento.

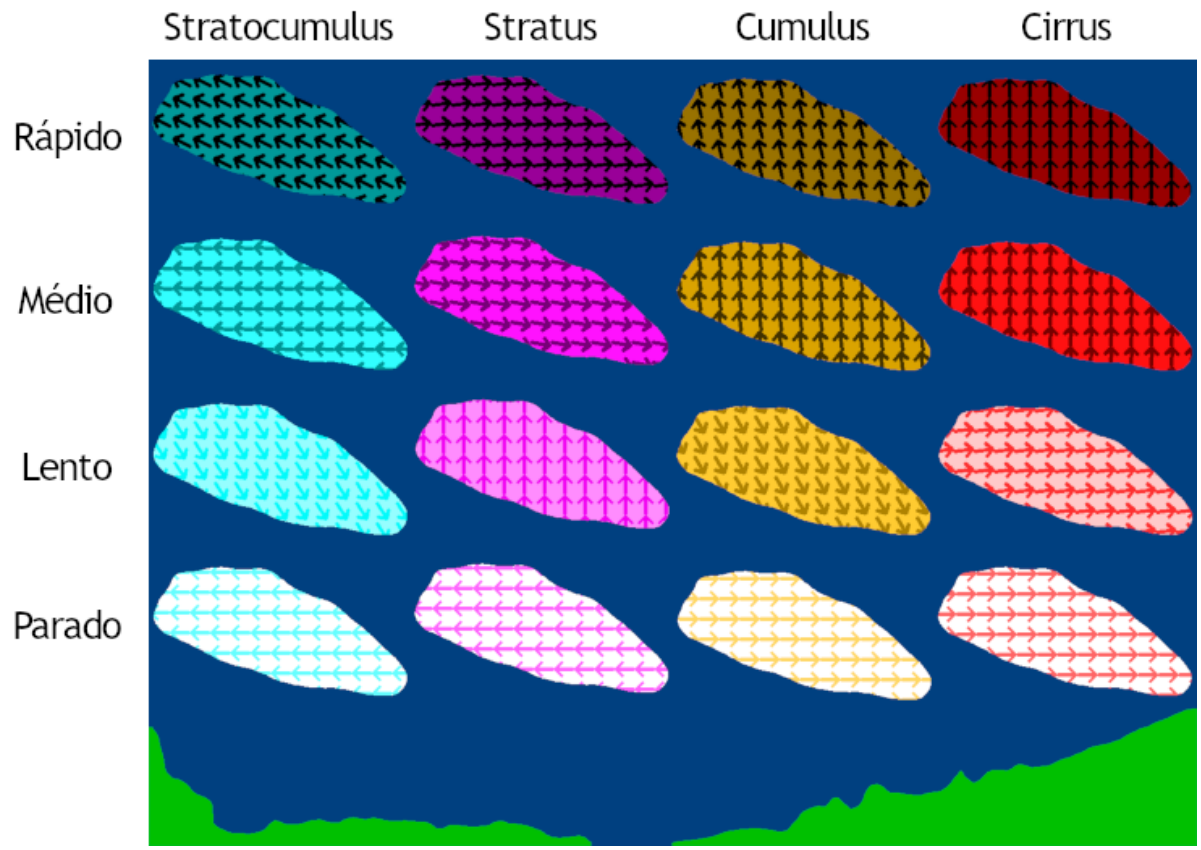
Na Figura 19, podemos ver um exemplo de vídeo produzido pelo *cloudseg.py* a partir de imagens reais. As três figuras à esquerda são uma sequência de imagens do horizonte capturadas em intervalos de 1 minuto, enquanto as imagens à direita são os *frames* do vídeo produzido, e correspondem a cada uma das imagens. Pode-se observar que os *frames* do vídeo identificam corretamente as nuvens estratocumuliformes e cirriformes, e que a direção das setas e luminosidade das nuvens indicam com certo grau de precisão a movimentação das nuvens.

Importante observar que, no último *frame* do vídeo, as nuvens não têm setas e apresentam luminosidade média. Isto ocorre porque o *script* não calcula vetor de deslocamento para a última imagem de uma sequência.

3.2.6 Funcionamento do *script*

O *script* *cloudseg.py* têm a função de unificar todas as etapas descritas acima para gerar vídeos descrevendo as classes e o movimento das nuvens presentes em uma

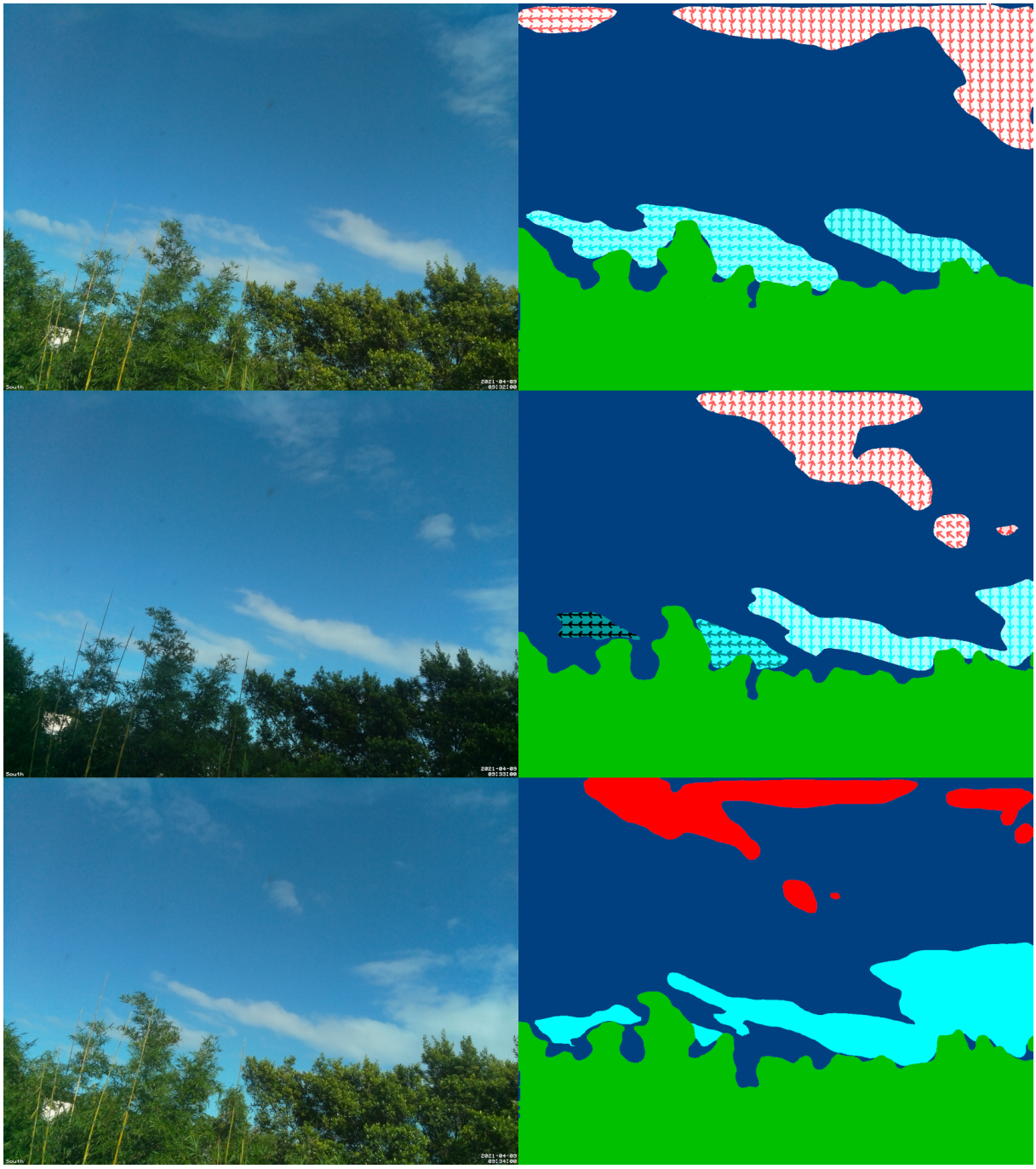
Figura 18 – Exemplo de representação de cores de imagens de nuvens produzidas pelo *cloudseg.py*.



Fonte: Elaborado pelo autor (2023).

sequência temporal de imagens do horizonte. Para isso, ele recebe como entrada a sequência de imagens e executa sobre ela, respectivamente, as etapas de segmentação semântica, correção de erros de segmentação, rastreamento de movimento e geração de vídeo. É possível configurar o formato de arquivo, a velocidade de exibição dos *frames*, dentre outras opções do vídeo. No Quadro 3, segue lista de argumentos e de funcionalidades do *script*.

Figura 19 – Sequência de três imagens do céu (à esquerda) e *frames* do vídeo gerado a partir das imagens (à direita).



Fonte: Elaborado pelo autor (2023).

Quadro 3 – Argumentos do *script cloudseg.py*

Nome	**Abreviação**	**Descrição**
–help	-h	Imprime mensagem de ajuda.
–dataset_dir	-d	Diretório que contém a sequência de imagens de entrada do *script*. Para a execução correta do *script*, o diretório deve conter somente arquivos de imagem.
–inf_out_dir	-i	Diretório onde serão salvas as imagens geradas pelo segmentador semântico, usadas para geração do vídeo.
–run_segmentation	-s	Se esta *flag* for usada, o *script* fará segmentação semântica. Senão, ele irá considerar que as imagens já foram segmentadas, e tentará obte-las do diretório indicado pelo argumento **-i**.
–ps_root_dir	-p	Diretório raiz do PaddleSeg.
–model_yaml	-m	Caminho completo para arquivo YAML do segmentador semântico.
–video_file_path	-v	Caminho completo para arquivo de vídeo de saída. Aqui pode ser escolhido o nome e extensão do arquivo. Se a *flag -g* estiver ativa, este argumento será interpretado como sendo o caminho para o diretório das imagens de saída.
–video_frame_dur	-f	Duração, em segundos, de exposição de cada *frame* do vídeo.
–only_annotation	-o	Se esta *flag* for usada, o vídeo gerado mostrará apenas as anotações das nuvens. Caso contrário, será gerado um vídeo com a imagem original e as anotações sobrescrevendo as nuvens, com grau de transparência configurado no argumento **-a**.
–annotation_alpha	-a	Grau de transparência das anotações. Pode assumir valores de 0.0 (anotações opacas) até 1.0 (totalmente transparentes). Não tem efeito quando a *flag* **-o** for usada.
–fourcc_string	-c	Código FourCC do *codec* usado para gerar o vídeo.
–generate_images	-g	Se esta *flag* for usada, a saída do *script* será uma sequência de imagens *png* em vez de um vídeo.

Fonte: Elaborada pelo autor (2023).

4 RESULTADOS E DISCUSSÃO

A seguir, serão apresentados os resultados obtidos pela solução após sua implementação. Na Seção 4.1, é analisada a qualidade da classificação realizada pelo segmentador semântico utilizando métricas de avaliação em *machine learning*, enquanto na Seção 4.2 são abordados problemas conhecidos dos vídeos gerados pelo *script*.

4.1 AVALIAÇÃO DE DESEMPENHO DO MODELO

A avaliação do modelo de segmentação semântica usado na solução foi feita logo após o seu treinamento, através do *script* **val.py**, fornecido pelo *framework* PaddleSeg. O *script* foi executado sobre os dados de teste, que correspondem a 10% do total de imagens no *dataset*, conforme visto na Seção 3.2.2. O resultado da execução do *script* mostra o tempo de execução das predições e traz uma série de métricas de avaliação de desempenho em *machine learning*. O tempo médio de segmentação de imagem foi de 174ms, enquanto o resultado das métricas calculadas pelo *script* é apresentado nas tabelas a seguir. A Tabela 1 traz o resultado de métricas calculadas sobre os resultados como um todo, enquanto a Tabela 2 traz resultados calculados por classe.

Tabela 1 – Resultados da avaliação do modelo

Métrica	**Valor**
mIoU	0,6580
Acurácia	0,8698
Coeficiente Kappa	0,8245
F1-Score	0,7752

Fonte: Elaborada pelo autor (2023).

Tabela 2 – Resultados da avaliação do modelo (por classe)

	Céu	***Stratocumulus***	***Stratus***	***Cirrus***	***Cumulus***	**Árvore**
IoU	0,7826	0,7788	0,5047	0,5855	0,3447	0,9517
Precisão	0,8751	0,8585	0,6983	0,7392	0,8639	0,9689
Recall	0,8811	0,8935	0,6454	0,7379	0,3645	0,9817

Fonte: Elaborada pelo autor (2023).

Examinando a Tabela 1, percebe-se que a acurácia geral do modelo é relativamente alta (0,8698), indicando que o modelo classifica os *pixels* corretamente na maioria dos casos. Além disso, o Kappa de 0,8245 indica uma concordância substancial entre as previsões do modelo e as classes reais. Entretanto, os valores mais baixos do F1-Score e do mIoU (Intersecção sobre união média) indicam uma dificuldade em segmentar as áreas das nuvens corretamente. Corrobora com esta hipótese o fato de o mIoU, que penaliza mais fortemente imprecisões na segmentação, ter resultado em um valor menor do que o F-Score.

Um dos fatores que diminui o mIoU é o IoU (Intersecção sobre união) para a classe das nuvens cumuliformes, que é de apenas 0,3447. Este IoU baixo, em conjunto com o *Recall* baixo, de apenas 0,3645, indicam que o modelo está confundindo muitas nuvens cumuliformes com outras classes. Uma explicação plausível para isso seria o fato de que um percentual muito baixo, de apenas 9,04%, das imagens de treinamento possuem nuvens do tipo cumuliforme. Por outro lado, as métricas para a classe estratocumuliforme, que aparece em 81,53% das imagens do *dataset*, foram todas acima de 0,75. A classe céu, que corresponde aos *pixels* não anotados das imagens originais e, portanto, está virtualmente presente em todas as imagens, apresentou resultados parecidos com o da classe estratocumuliforme. Tudo isso nos indica que a presença da classe em um número significativo de imagens do *dataset* pode ser crucial para que se identifique mais nuvens desta classe corretamente.

É importante destacar que a classe Árvore teve desempenho excepcional, apresentando valor acima de 0,95 para todas as métricas. Isto se deve tanto ao fato de a classe estar presente em 99,3% dos dados de treinamento quanto ao fato destas serem imóveis e com posição fácil de predizer. As árvores, muitas vezes, ocupam cerca de 1/4 da imagem, e se mantém praticamente na mesma posição em todas as imagens de uma sequência de imagens do mesmo lugar, o que facilita ao modelo prever a localização dos *pixels* de árvore. Tais valores acima da média, aliados à grande quantidade de *pixels* que as árvores costumam ocupar em uma imagem, fazem subir o valor de métricas como o mIoU, acurácia, o F1-Score e o coeficiente Kappa.

4.2 PROBLEMAS CONHECIDOS NA REPRESENTAÇÃO DE NUVENS

Durante a análise qualitativa dos vídeos gerados pelo *script* desenvolvido, observaram-se determinados problemas na classificação e na representação do rastreamento do movimento das nuvens, problemas estes decorrentes de medidas corretivas adotadas pelo presente trabalho na abordagem de outras dificuldades conhecidas. As subseções a seguir trazem exemplos e detalham tais problemas, onde também será feita a análise de suas causas.

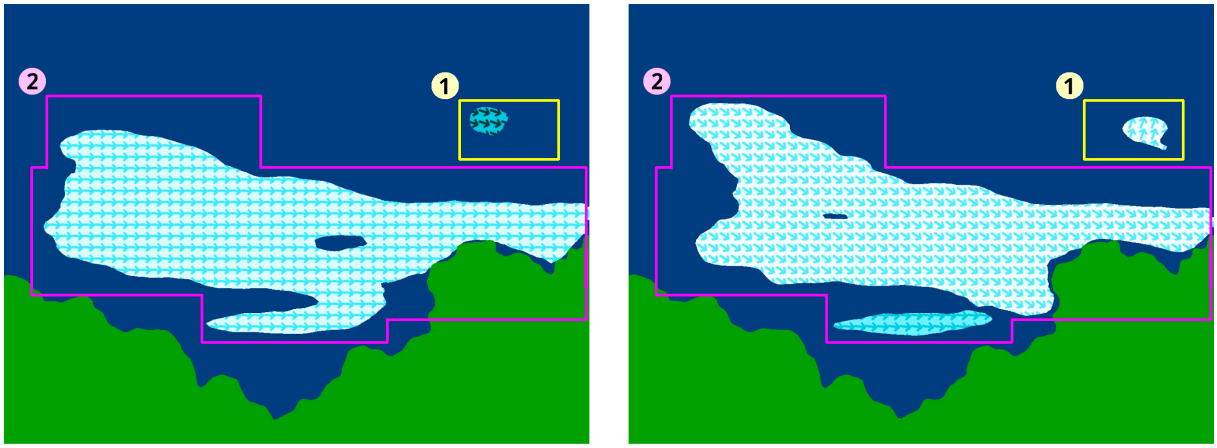
4.2.1 Indicação imprecisa da direção de movimento

Ao observar de forma mais minuciosa as nuvens de cada *frame* dos vídeos gerado pelo *script*, é possível encontrar situações onde as setas indicativas de direção de movimento de nuvem não correspondem precisamente ao deslocamento real da nuvem. Isto ocorre principalmente em nuvens que, na passagem de um *frame* a outro, mudam de tamanho e forma. Normalmente, a magnitude do vetor resultante de deslocamento deste tipo de nuvem é pequeno e, portanto, a nuvem é representada em cores claras.

Um exemplo típico da situação descrita acima pode ser vista na Figura 20, que

mostra dois *frames* consecutivos de um vídeo gerado a partir de uma das sequências de imagens do céu presentes no *dataset* utilizado no presente trabalho. No exemplo, onde o *frame B* é subsequente ao *frame A*, pode-se observar o comportamento de duas nuvens e as respectivas indicações de movimento.

Figura 20 – Sequência obtida do vídeo gerado pelo *cloudseg.py* onde ocorre imprecisão na direção apontada pelas setas. Na figura, o *frame B* é subsequente ao *frame A*. A imprecisão ocorre na nuvem *2*.



Fonte: Elaborada pelo autor (2023).

Na Figura 20, observa-se que na nuvem 1, que se desloca em distância relativamente significativa para o canto inferior direito da imagem, a representação do vetor de deslocamento está correta, retratada com cor escura para indicar a magnitude do deslocamento, e com setas apontando para a direção do movimento. Já a nuvem 2 é um exemplo característico de representação imprecisa: é uma nuvem clara (e, portanto, com magnitude baixa de deslocamento) onde setas indicam movimento para a direita, mas onde a nuvem não se move realmente para a direita, sofrendo distorções de forma e divisões em mais nuvens em vez disso.

É importante destacar que, em partes, o problema descrito é relacionado à forma de representação gráfica do vetor de deslocamento, que pode causar confusões. Como as nuvens dos vídeos gerados pelo *script* exibem setas mesmo que estejam totalmente paradas, existe a possibilidade de, por exemplo, uma nuvem totalmente branca (portanto, imóvel) que apresenta setas para a direita ser interpretada como se houvesse deslocamento à direita. Entretanto, como não seria possível identificar a qual classe pertence uma nuvem totalmente branca sem setas, optou-se por manter setas coloridas de acordo com classificação mesmo em nuvens imóveis.

Por outro lado, sabe-se que o cálculo do vetor de deslocamento da nuvem possui imprecisões inerentes. Uma delas diz respeito ao funcionamento dos algoritmos de fluxo óptico em geral, que partem da premissa de que não há mudança na intensidade dos

pixels de um *frame* a outro da sequência de imagens analisada. Esta hipótese dificilmente seria verdadeira para sequências de imagens do céu no horizonte, onde há oscilação de luminosidade a todo tempo devido a condições climáticas. A partir disso, podemos deduzir que a hipótese da intensidade constante também não se faria verdadeira para as imagens usadas para o cálculo dos vetores de deslocamento, que seriam as imagens obtidas após a segmentação semântica das imagens do céu.

Outra imprecisão inerente ao cálculo do vetor de deslocamento se deve ao fato de este ser um vetor predominante e, portanto, uma aproximação obtida a partir do deslocamento individual de cada um dos *pixels* da nuvem. Assim, em situações onde há *pixels* se movendo em várias direções, como em nuvens que se expandem, contraem, ou sofrem algum tipo de metamorfose, as setas não necessariamente indicam o movimento real da nuvem, pois a média dos vetores (que provavelmente será baixa) pode não corresponder à transformação sofrida por ela. Da mesma forma, uma nuvem que se desloca um pouco para um lado, mas se expande para outras direções, pode ter os vetores dos *pixels* que caracterizam o movimento da nuvem ofuscados pelos vetores dos *pixels* da expansão durante o cálculo, resultando em um vetor de movimento impreciso.

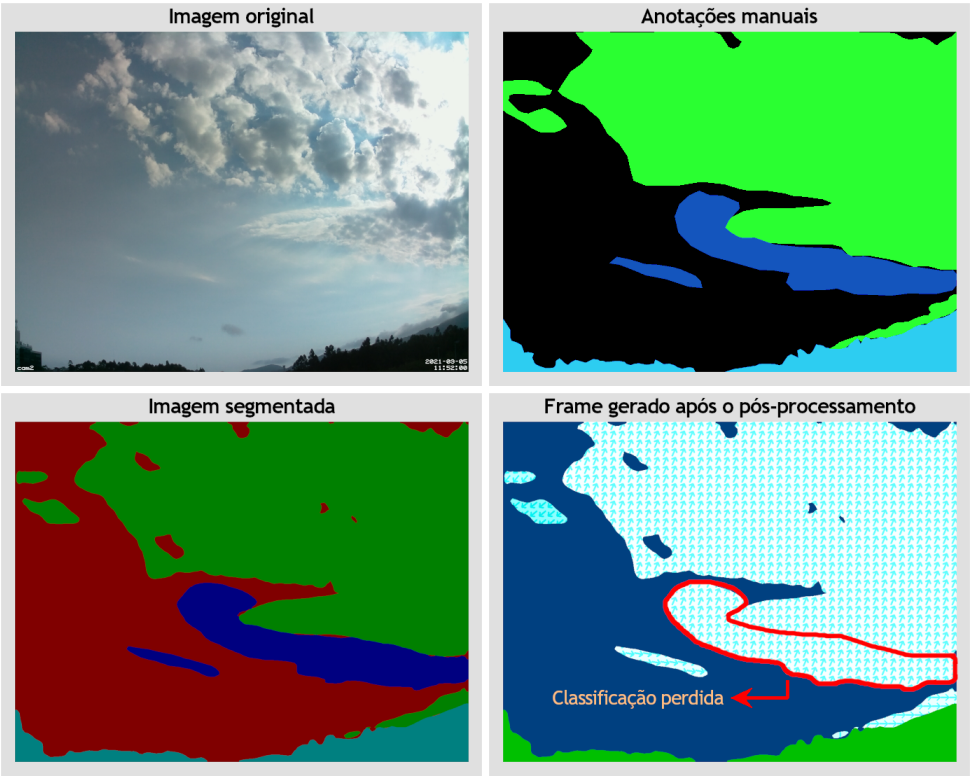
4.2.2 Reclassificação indesejada no pós-processamento

O pós-processamento de imagens inferidas pelo segmentador semântico, apresentado na subseção 3.2.3, embora permita corrigir certos problemas de classificação de nuvens, traz em seu uso alguns efeitos colaterais. O uso de componentes conexas e de imagens de *frames* anteriores para identificação da classe a qual uma nuvem pertence, em situações específicas, podem acabar reclassificando nuvens corretamente classificadas. As duas situações conhecidas onde isto ocorre serão detalhadas abaixo.

A primeira situação ocorre quando nuvens de classe diferentes estão muito próximas uma da outra, ou quando uma sobrepõe a outra. Quando isto acontece, devido ao fato de o algoritmo de pós-processamento buscar componentes conexas na imagem para identificação de nuvens, as duas nuvens próximas acabam sendo identificadas como uma só. Como o algoritmo atribui uma única classe a todos os *pixels* de uma componente conexa, ele atribuirá a nuvem menor a mesma classificação atribuída à nuvem maior. Ou seja, quando há nuvens contíguas ou sobrepostas, a nuvem menor será identificada como sendo da mesma classe que a nuvem maior, mesmo que este não seja o caso. A Figura 21 ilustra a situação.

O segundo problema surge quando duas nuvens de classes diferentes e em *frames* consecutivos de uma sequência de imagens ocupam *pixels* nas mesmas coordenadas x e y em suas respectivas imagens. Este tipo de situação é problemática porque o cálculo da classe predominante feito pelo algoritmo leva em consideração tanto a imagem que está sendo pós-processada quando os dois *frames* anteriores. Em casos onde há mudanças muito bruscas no posicionamento das nuvens de um *frame* a outro, é possível que uma nuvem de

Figura 21 – Exemplo de problema onde o pós-processamento faz a nuvem menor assumir a classe de uma nuvem maior contígua a ela.

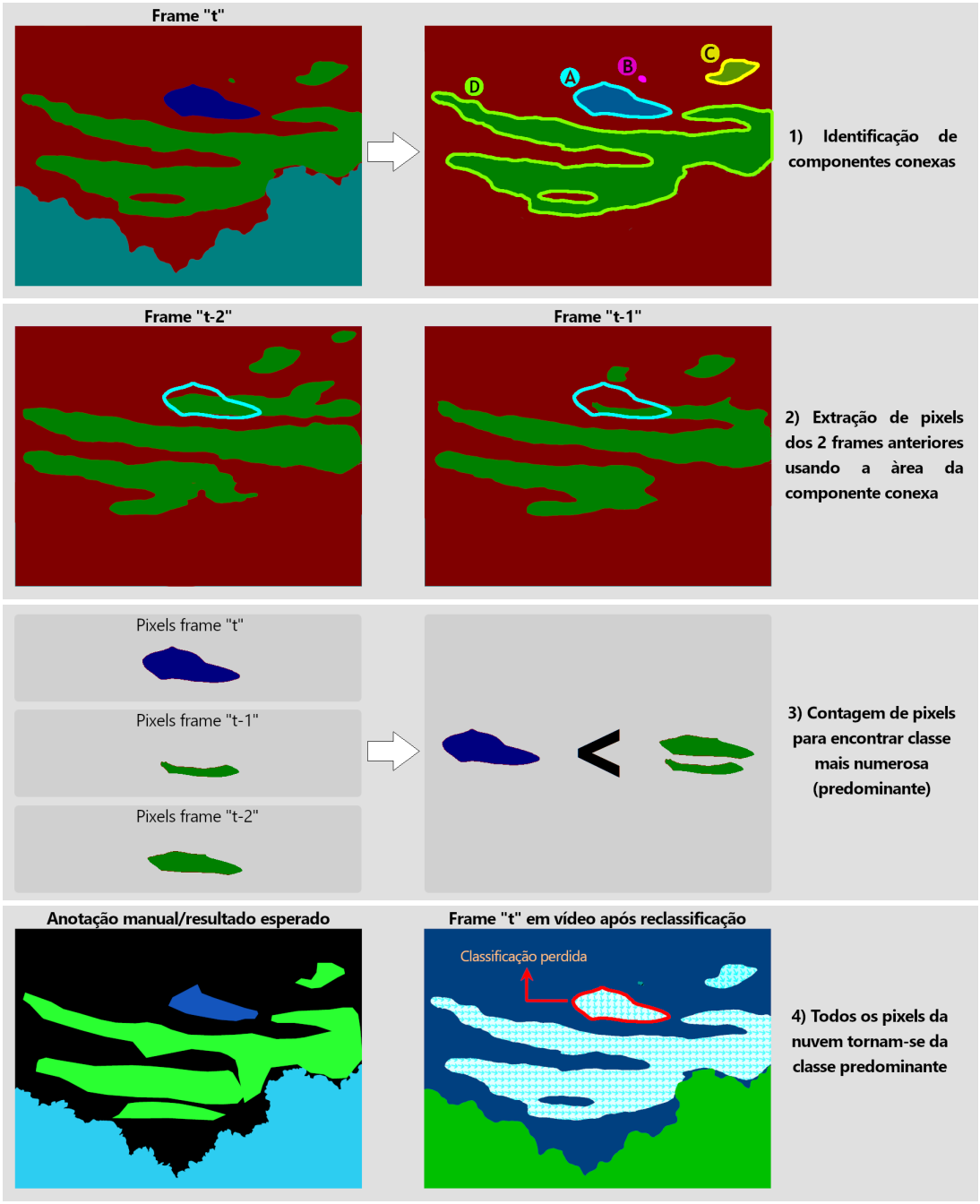


Fonte: Elaborada pelo autor (2023).

determinada classe ocupe um lugar onde, nos dois *frames* anteriores, havia uma nuvem de outra classe. Nesses casos, a contagem de *pixels* do algoritmo de pós-processamento encontrará um número maior de *pixels* da classe da nuvem dos *frames* anteriores do que da classe da nuvem do *frame* em pós-processamento. Tal comportamento do algoritmo não é desejado, pois seu objetivo é o de impedir que uma nuvem mude de classificação de um *frame* a outro, e não de fazê-la assumir a classificação de outras nuvens de *frames* anteriores. A Figura 22 ilustra o problema.

Apesar dos problemas identificados durante a análise, foi decidido manter o algoritmo de pós-processamento. Essa escolha baseou-se na consideração de que cometer um erro ao afirmar que uma nuvem tem a mesma classificação que a do *frame* anterior é preferível a cometer um erro ao afirmar que ela pertence a uma classe diferente quando, na realidade, é a mesma. A justificativa para essa decisão reside no comportamento do algoritmo de fluxo ótico, que funciona através da detecção do deslocamento de *pixels* de mesma intensidade, ou seja, da mesma classe. Dessa forma, ao cometer um erro ao afirmar que uma nuvem pertence a uma classe diferente, não apenas ocorre um equívoco na classificação, mas também perdemos o rastreamento do seu movimento. Por outro lado, ao cometer um erro ao afirmar que a nuvem é da mesma classe, apenas a classificação é

Figura 22 – Exemplo de problema onde pós-processamento de nuvens em *frames* anteriores forçam uma reclassificação indesejada na nuvem A.



Fonte: Elaborada pelo autor (2023).

equivocada, e o rastreamento do movimento permanece intacto. Essa abordagem visa a otimização do rastreamento de nuvens, mesmo em situações de classificação incorreta.

5 CONCLUSÃO

Através da solução experimental implementada no presente trabalho, foi possível demonstrar a viabilidade de se efetuar classificação e rastreio de movimentação de nuvens em imagens do horizonte, empregando o modelo de rede neural em nível do estado-da-arte PP-LiteSeg para a tarefa de segmentação semântica, e o método de fluxo óptico denso de Gunnar-Farneback para identificar o movimento das nuvens. O *script cloudseg.py*, desenvolvido neste trabalho, integra o segmentador semântico e o algoritmo de detecção de fluxo óptico, sendo capaz assim de, a partir de uma sequência temporal de imagens do céu, gerar um vídeo onde as nuvens aparecem segmentadas e com cores e setas indicando a magnitude e a direção do seu deslocamento.

O modelo de segmentação semântica utilizado na solução desenvolvida usou em seu treinamento o *dataset* Clouds-1000, que possui 1000 imagens de imagens do céu, com classificação de nuvens presentes em anotações realizadas por uma equipe multidisciplinar e revisadas por meteorologista sêniores. Apesar de haver significativo desbalanceamento de dados no *dataset* utilizado no treino, o modelo atingiu resultados satisfatórios, atingindo uma acurácia de 0,8698%. Ao analisar o IoU, a acurácia e o *recall* das predições por classe, percebemos que o modelo consegue distinguir muito bem o céu e as árvores das nuvens, apresentando resultados acima de 0,75 em todas as métricas. Entretanto, há discrepância na capacidade de identificação de nuvens de cada classe, de modo que nuvens do tipo estratocumuliforme apresentam métricas com bons resultados nas métricas, nuvens do tipo estratiforme e cirriforme apresentam resultados medianos, enquanto nuvens do tipo cumuliforme apresentam IoU e *recall* próximos a 0,35.

Para cobrir as deficiências do modelo, implementou-se uma etapa de pós-processamento do resultado da segmentação semântica, executada antes deste resultado ser usado pelo algoritmo de fluxo óptico para o rastreio do movimento das nuvens. Tal etapa fez-se necessária devido a problemas de segmentação que se manifestaram ocasionalmente, como a fragmentação de uma única nuvem em várias classes e a mudança súbita de classe de uma nuvem de um *frame* da sequência de imagens a outro. O algoritmo de pós-processamento suplanta estes problemas fazendo uma reclassificação de nuvens, que considera, durante seus cálculos, as componentes conexas da imagem em tratamento e a classificação de *pixels* nas duas imagens anteriores.

O rastreamento do movimento das nuvens foi feito usando a saída do algoritmo de pós-processamento da segmentação de imagens, sob os quais foi calculado o fluxo óptico denso pelo método de Gunnar-Farneback entre cada par de imagens subsequentes. A partir dos vetores de deslocamento de *pixels* obtidos pelo algoritmo de Gunnar-Farneback, o *script* desenvolvido no presente trabalho calcula um vetor predominante para cada nuvem, usando, neste cálculo, os vetores dos *pixels* da nuvem em questão. Os vetores predominantes calculados mostraram-se capazes de fornecer uma estimativa razoável da

velocidade e da direção do movimento da nuvem, embora apresentem imprecisões em algumas situações, como nos casos de nuvens sofrendo distorções como expansão, retração ou divisões em mais nuvens.

Em resumo, a realização da tarefa de classificação e predição de movimento de nuvens usando o segmentador semântico PP-LiteSeg e o método de fluxo óptico de Gunnar-Farneback mostrou-se viável, apesar das limitações dos dados de treinamento disponíveis e embora ainda caibam ajustes para correções de algumas imprecisões.

6 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

Durante a análise dos resultados obtidos, algumas recomendações para trabalhos futuros emergiram. No que diz respeito à segmentação semântica, embora o desempenho tenha sido satisfatório para separar nuvens do céu, identificar classes específicas mostrou-se desafiador, com apenas nuvens do tipo estratocumuliforme apresentando resultados realmente favoráveis. Como as investigações feitas sobre os dados utilizados e as métricas obtidas no treinamento do modelo demostraram que a capacidade de classificar nuvens de certo tipo é proporcional à quantidade de nuvens deste tipo disponíveis no *dataset*, há razões suficientes para acreditar que um melhor balanceamento de classes no conjunto de dados levaria a resultados melhores na tarefa de classificação de nuvens de qualquer tipo. Estudos subsequentes que utilizassem um *dataset* mais completo e balanceado no treinamento do modelo provavelmente teriam mais sucesso na tarefa de segmentação semântica.

Propõe-se também a comparação do modelo PP-LiteSeg com outras arquiteturas de redes neurais e técnicas de *machine learning*, a fim de avaliar quais delas são as mais adequadas para a classificação de nuvens no horizonte. Para este fim, poderia-se aproveitar a flexibilidade do *script* desenvolvido no presente trabalho, tendo em vista que ele é capaz de fazer inferências utilizando qualquer modelo de rede neural exportado no formato aceito pelo *framework* PaddleSeg. Além disso, sugere-se a adaptação do *script* para a aceitação de modelos exportados para formatos abertos, como NNEF[1] e ONNX[2], permitindo assim que redes neurais de diferentes *frameworks* e motores de inferência de *deep learning* possam ser executados. Tal abordagem não só amplia o leque de opções de modelos aceitos pelo *script* como viabiliza comparações justas entre modelos de diferentes *frameworks*.

Outra abordagem a qual estudos posteriores onde se pretenda adotar o mesmo modelo empregado no presente trabalho poderiam utilizar para obter melhores resultados na segmentação semântica seria investir em um ajuste mais criterioso dos hiperparâmetros de treinamento, já que isto não foi possível no presente trabalho devido a limitações no *hardware* disponível.

No contexto do pós-processamento, que se revelou necessário devido a problemas específicos de segmentação, sugere-se que treinar o modelo com dados melhores pode eliminar a necessidade dessa etapa. Isso incluiria a utilização de dados que abordem os problemas específicos encontrados. O problema da mudança de classes de um *frame* a outro, por exemplo, poderia ser mitigado ou superado aumentando-se o desempenho geral do modelo na identificação de nuvens, através das melhorias no *dataset* propostas anteriormente. Já o problema da fragmentação de nuvens em várias classes poderia ser contornado através da inclusão de imagens ambíguas no *dataset*, contendo tanto nuvens sobrepostas e contíguas quanto nuvens grandes que se assemelham a nuvens sobrepostas, com ano-

[1] https://www.khronos.org/nnef
[2] https://onnx.ai/

tações humanas desfazendo estas ambiguidades. Usar dados com tais características no treinamento do modelo pode ser uma estratégia para prepará-lo para classificar nuvens corretamente neste tipo de situação.

Além disso, considera-se que mudanças no algoritmo de cálculo de rastreamento de nuvens podem eliminar a necessidade de pós-processamento. Se considerarmos que as nuvens podem se transformar, por exemplo, a mudança de classes da nuvem de um *frame* a outro deixaria de ser um problema. Porém, seriam necessárias alterações na forma que o rastreamento de nuvens é feito, pois, atualmente, este busca por *pixels* de mesma intensidade em *frames* consecutivos a fim de calcular o deslocamento. Uma alteração possível que futuros trabalhos poderiam considerar para atingir este objetivo seria a de desconsiderar a classe de nuvem no cálculo de fluxo óptico.

Quanto ao rastreio de movimento, que possui imprecisões conhecidas, existe a possibilidade de substituição do método de Gunnar-Farneback por outros métodos de fluxo óptico ou outras técnicas de registro de imagens para obter informações sobre o movimento das nuvens. Investigações futuras poderiam comparar o desempenho de diferentes técnicas de registro de imagem na tarefa de detectar movimento de nuvens, a fim de descobrir se alguma delas supera os problemas do método atual.

Outra abordagem para aprimorar o rastreamento da movimentação de nuvens, que permitiria manter o uso do método de Gunnar-Farneback e que têm potencial de superar as imprecisões intrínsecas ao cálculo de vetor predominante seria a de substituir o conceito de um único vetor por nuvem pela ideia de vetores predominantes por seção de nuvens. Esta estratégia consistiria em dividir a nuvem em seções, e usar o fluxo óptico dos *pixels* de cada uma elas para calcular um vetor predominante por seção. Assim, vários vetores de deslocamento estariam associados a uma única nuvem, permitindo uma compreensão mais precisa de movimentos como expansão ou encolhimento de nuvens, o que é impossível com apenas um vetor predominante por nuvem.

Uma terceira opção para melhorias nos rastreio seria explorar diferentes métodos de cálculo de vetor predominante que não se baseiem na soma de vetores de *pixels*.

Por último, seria relevante investigar estratégias para integrar a solução implementada em um sistema real que empregue *nowcasting* na estimativa de produção de energia solar. Desta forma, seria possível avaliar a efetiva viabilidade da aplicação da tecnologia desenvolvida em contextos práticos da área de fotovoltaica. Avaliações sobre o tempo de execução da segmentação semântica, do pós-processamento e do rastreio de movimento deverão ser consideradas nesses casos, dada a importância de fazer predições em tempo hábil quando é necessária previsão solar intra-hora.

REFERÊNCIAS

ARRAIS, Juliana Marian *et al.* **Systematic Literature Review on Ground-Based Cloud Tracking Methods for Nowcasting and Short-term Forecasting**. [*S.l.*], 2022. DOI: https://doi.org/10.13140/RG.2.2.17730.25281. Disponível em: https://www.researchgate.net/publication/364571948_Systematic_Literature_Review_on_Ground-Based_Cloud_Tracking_Methods_for_Nowcasting_and_Short-term_Forecasting. Acesso em: 8 jul. 2023.

BARRETT, Eric C.; GRANT, Colin K. **The identification of cloud types in LANDSAT MSS images**. [*S.l.*], 1976.

FARNEBÄCK, Gunnar. Two-Frame Motion Estimation Based on Polynomial Expansion. *In:* 2749. SCIA13 : Gothenburg, Sweden. [*S.l.*: *s.n.*], 2003. (Lecture Notes in Computer Science, 2749), p. 363–370.

HILL, Cody A. *et al.* Battery energy storage for enabling integration of distributed solar power generation. **IEEE Transactions on smart grid**, IEEE, v. 3, n. 2, p. 850–857, 2012. DOI: https://doi.org/10.1109/TSG.2012.2190113. Disponível em: https://www.researchgate.net/publication/260581959_Battery_Energy_Storage_for_Enabling_Integration_of_Distributed_Solar_Power_Generation. Acesso em: 8 jul. 2023.

INMAN, Rich H.; PEDRO, Hugo T. C.; COIMBRA, Carlos F. M. Solar forecasting methods for renewable energy integration. **Progress in energy and combustion science**, Elsevier, v. 39, n. 6, p. 535–576, 2013. DOI: https://doi.org/10.1016/j.pecs.2013.06.002. Disponível em: http://www.sciencedirect.com/science/article/pii/S0360128513000294. Acesso em: 8 jul. 2023.

KUMARI, Pratima; TOSHNIWAL, Durga. Deep learning models for solar irradiance forecasting: A comprehensive review. **Journal of Cleaner Production**, Elsevier, v. 318, p. 128566, 2021. DOI: https://doi.org/10.1016/j.jclepro.2021.128566. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0959652621027736. Acesso em: 11 jul. 2023.

LIN, Fan; ZHANG, Yao; WANG, Jianxue. Recent advances in intra-hour solar forecasting: A review of ground-based sky image methods. **International Journal of Forecasting**, v. 39, n. 1, p. 244–265, 2023. ISSN 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2021.11.002. Disponível em: https://www.sciencedirect.com/science/article/pii/S016920702100176X.

LUCAS, Bruce D.; KANADE, Takeo. An Iterative Image Registration Technique with an Application to Stereo Vision. *In:* PROCEEDINGS of the 7th International Joint Conference on Artificial Intelligence - Volume 2. Vancouver, BC, Canada: Morgan Kaufmann Publishers Inc., 1981. (IJCAI'81), p. 674–679.

MAHONY, Niall O' *et al.* Deep Learning vs. Traditional Computer Vision. **CoRR**, abs/1910.13796, 2019. arXiv: 1910.13796. Disponível em: http://arxiv.org/abs/1910.13796.

MARTINS, Bruno Juncklaus; CERENTINI, Allan *et al.* Systematic review of nowcasting approaches for solar energy production based upon ground-based cloud imaging. **Solar Energy Advances**, Elsevier, v. 2, p. 100019, 2022. DOI: https://doi.org/10.1016/j.seja.2022.100019. Disponível em: https://www.sciencedirect.com/science/article/pii/S2667113122000079. Acesso em: 11 jul. 2023.

MARTINS, Giuliano L.; NETO, Sylvio Luiz Mantelli; RÜTHER, Ricardo. Evaluating the performance of radiometers for solar overirradiance events. **Solar Energy**, Elsevier, v. 231, p. 47–56, 2022. DOI: https://doi.org/10.1016/j.solener.2021.11.050. Disponível em: https://www.sciencedirect.com/science/article/pii/S0038092X21010100. Acesso em: 11 jul. 2023.

NASCIMENTO, Lucas Rafael do; BRAGA, Marília *et al.* Performance assessment of solar photovoltaic technologies under different climatic conditions in Brazil. **Renewable Energy**, Elsevier, v. 146, p. 1070–1082, 2020. DOI: https://doi.org/10.1016/j.renene.2019.06.160. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0960148119310006. Acesso em: 11 jul. 2023.

NASCIMENTO, Lucas Rafael do; SOUZA VIANA, Trajano de *et al.* Extreme solar overirradiance events: Occurrence and impacts on utility-scale photovoltaic power plants in Brazil. **Solar Energy**, Elsevier, v. 186, p. 370–381, 2019. DOI: https://doi.org/10.1016/j.solener.2019.05.008. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0038092X19304530. Acesso em: 11 jul. 2023.

NETO, Sylvio Luiz Mantelli *et al.* Hierarchical color similarity metrics for step-wise application on sky monitoring surface cameras. **Authorea Preprints**, Authorea, 2022. DOI: https://doi.org/10.1002/essoar.10503135.1. Disponível em: https://essopenarchive.org/doi/full/10.1002/essoar.10503135.1. Acesso em: 11 jul. 2023.

PENG, Juncai *et al.* **PP-LiteSeg: A Superior Real-Time Semantic Segmentation Model**. [*S.l.*: *s.n.*], 2022. arXiv: 2204.02681 [cs.CV].

Apêndices

APÊNDICE A – ARTIGO

Classificação e Rastreamento de Nuvens no Horizonte para Aplicações *Nowcasting* via Segmentação Semântica e Estimativa de Fluxo Ótico

Gilberto P. Ricci Neto[1]

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina
88.040-900 – Florianópolis – SC – Brasil

gilberto.ricci@grad.ufsc.br

Abstract. *This present work proposes an approach using deep learning to predict cloud movement and enhance predictability in solar energy production. The PP-LiteSeg model is employed for semantic segmentation of clouds, while the Gunnar Farneback optical flow method is used to track cloud movement. The results show good accuracy in segmentation and coherence in predicting cloud displacement. The Python script developed in the study demonstrates the feasibility of the solution, displaying cloud segmentation and movement in a temporal sequence of images through a video.*

Resumo. *O presente trabalho propõe uma abordagem utilizando deep learning para prever o movimento de nuvens e melhorar a previsibilidade na produção de energia solar. O modelo PP-LiteSeg é empregado para segmentação semântica das nuvens, enquanto o método de fluxo óptico de Gunnar Farneback é utilizado para rastrear o movimento das nuvens. Os resultados mostram boa acurácia na segmentação e coerência na previsão do deslocamento das nuvens. O script em Python desenvolvido no trabalho demonstra a viabilidade da solução, exibindo a segmentação e o deslocamento das nuvens em uma sequência temporal de imagens por meio de um vídeo.*

1. Introdução

A necessidade de reduzir a dependência de matrizes energéticas poluentes, como os combustíveis fósseis, na produção de energia elétrica, impulsiona a adoção de tecnologias renováveis, especialmente a energia solar [Kumari and Toshniwal 2021]. O declínio nos custos dos equipamentos fotovoltaicos e a abundância solar global destacam-se entre os benefícios [Kumari and Toshniwal 2021, do Nascimento et al. 2019]. Apesar da eficiência na geração de eletricidade por meio de sistemas fotovoltaicos conectados à rede elétrica [do Nascimento et al. 2019], a intermitência da irradiância solar, influenciada por condições climáticas, é um desafio [Martins et al. 2022]. A movimentação de nuvens pode resultar em quedas abruptas na potência de saída, causando variações na voltagem e impactando negativamente equipamentos da rede elétrica ligada ao sistema fotovoltaico [Hill et al. 2012]. Esses desafios exigem, muitas vezes, o uso de geradores complementares, aumentando os custos dos sistemas fotovoltaicos [Inman et al. 2013].

Neste cenário onde a volatilidade da produção de energia fotovoltaica é um entrave, formas de antecipar a quantidade de irradiação que incidirá sobre os painéis solares são desejáveis, e uma forma eficaz de fazê-lo é empregar técnicas de previsão de tempo.

Como as oscilações na irradiância podem ocorrer em questão de segundos, as técnicas de previsão solar com o menor horizonte de predição são as mais adequadas para a tarefa. Assim, diversas formas de *nowcasting* (previsão intra-hora), capazes de antecipar irradiação de minutos à frente, vêm sido propostos para melhorar o desempenho de plantas fotovoltaicas. Dentre os métodos existentes, os que fazem predição solar usando imagens de nuvens no céu se destacam [Lin et al. 2023].

O presente trabalho propõe uma solução automatizada para prever o movimento de nuvens em imagens do céu. O processo envolve a segmentação semântica de nuvens e a estimativa de direção e velocidade de deslocamento. Utilizando fotografias do horizonte, o projeto evita problemas associados ao uso de câmeras olho-de-peixe apontadas para o zênite, como alto custo e distorções de imagem [Arrais et al. 2022]. A segmentação semântica é feita por um modelo de rede neural no estado-da-arte, o PP-LiteSeg, treinado com um *dataset* de imagens de nuvens no horizonte. Já a predição de movimento é feita através do método de fluxo óptico denso de Gunnar-Farneback, que apresenta vantagens sobre métodos de fluxo óptico esparso no cálculo de deslocamento de nuvens. A solução, implementada em Python, recebe uma sequência temporal de imagens de nuvens no horizonte como entrada e, como saída, retorna um vídeo com segmentação e indicação de movimento das nuvens.

2. Trabalhos Relacionados

A revisão sistemática de [Lin et al. 2023] propõe um *framework* pra descrever características comuns de trabalhos envolvendo predição solar intra-hora. O *framework* desenvolvido no trabalho em questão consiste em quatro módulos: o módulo 1 trata da captura de imagens do céu, o módulo 2 trata do pré-processamento das imagens para predição, o módulo 3 trata da predição do movimento das nuvens e o módulo 4 fornece os resultados finais da previsão de irradiância solar ou da saída de energia solar.

No que diz respeito ao módulo 1, de captura de imagens, verificou-se que a maioria dos trabalhos relacionados usa câmeras semi-profissionais com lente olho-de-peixe, capturando fotos do zênite [Arrais et al. 2022]. Este tipo de câmera é cara, e sua lente distorce o formato das nuvens, exigindo pre-processamento de imagem antes da predição [Arrais et al. 2022]. No presente trabalho, optou-se pelo uso de fotos do horizonte capturadas com câmeras convencionais no rastreamento de movimento de nuvens. Tal abordagem evita o custo elevado das câmeras semi-profissionais, e dispensa o pré-processamento para correção de distorções.

A solução implementada no presente trabalho corresponde ao módulo 3 do *framework* descrito. Este módulo é subdividido em 2 submódulos: o *cloud recognition*, onde usa-se segmentação semântica para identificar as nuvens, e o *cloud motion modeling*, onde usam-se técnicas de registro de imagens para prever o movimento das nuvens [Lin et al. 2023]. Para o primeiro submódulo, 71,9% dos trabalhos relacionados usa métodos clássicos, enquanto, no presente trabalho, usa-se *deep learning*, uma abordagem mais moderna [Arrais et al. 2022]. Já para segundo submódulo, a abordagem mais comum é o método de fluxo óptico esparso de Lucas-Kanade [Arrais et al. 2022]. Como métodos de fluxo óptico esparso dependem de cantos e bordas em objetos, cantos e bordas estes que não são facilmente encontrados em objetos disformes como nuvens [Arrais et al. 2022], optou-se, no presente trabalho, por usar o método de fluxo óptico

denso de Gunnar-Farneback para a modelagem de movimento de nuvens.

3. Metodologia

Esta seção irá trazer detalhes sobre o conjunto de dados utilizado para o treinamento, sobre o uso do PP-LiteSeg na segmentação semântica de nuvens, sobre o cálculo do movimento das nuvens através do método de Gunnar-Farneback e sobre a geração do vídeo que demonstra o rastreamento de movimento de nuvens.

3.1. Conjunto de dados

O *dataset* utilizado no treinamento do modelo foi o Clouds-1000, criado pelo LAPIX (Laboratório de Processamento de Imagens e Computação Gráfica da UFSC). O Clouds-1000 é composto por um conjunto de 1000 imagens do céu no horizonte, juntamente com anotações que segmentam e classificam as nuvens. As anotações foram feitas manualmente por analistas de dados no Supervisely, uma ferramenta *web* para *machine learning*, anotações estas que foram revisadas por meteorologistas para garantir sua corretude.

Para a captura das imagens, foi escolhida uma área próxima à UFSC, com boa visibilidade do céu. As coordenadas geográficas do local são 27°36'28.1"S 48°30'48.5"W. As imagens do *dataset* foram capturadas entre 21 março de 2021 a 21 de junho de 2021. O equipamento utilizado foram duas câmeras Raspberry Pi modelo V1.3, apontando para as direções norte e sul. Através do o sistema operacional personalizado Nimbus Gazer, as câmeras foram configuradas para capturar imagens em uma taxa de 1 *frame* por minuto.

A classificação de nuvens das anotações utiliza as famílias propostas pelo trabalho de [Barrett and Grant 1976], que são agrupamentos dos gêneros de nuvem da WMO (World Meteorological Organization), cujo critério e a textura de nuvem. A Tabela 1 apresenta as classes presentes no *dataset*, e a quantidade de imagens nas quais cada classe está presente. Nota-se que, além das classes que correspondem às famílias de nuvem, ha também a classe Árvore, que representa não só árvores como também outros elementos que obstruem a visão do céu, como prédios, postes de luz, etc.

Tabela 1. Quantidade de nuvens de cada família presentes no *dataset*.

Classe	Quantidade de imagens	Quantidade de imagens (%)
Árvore	989	99,3
Estratocumuliforme	812	81,53
Estratiforme	271	27,21
Cirriforme	285	28,61
Cumuliforme	90	9,04
Cumulonimbiforme	0	0,00

3.2. Segmentação Semântica

O modelo de rede neural escolhido para a segmentação semântica de nuvens foi o PP-LiteSeg. Escolheu-se este modelo devido aos resultados no estado da arte alcançados por ele, conforme descrito em seu artigo original, onde verifica-se que este apresenta bom equilíbrio entre velocidade de inferência e intersecção sobre união média, em testes realizados com vários modelos diferentes de segmentação semântica, nos *datasets* CamVid e Cityscapes [Peng et al. 2022].

O PP-LiteSeg adota uma arquitetura *encoder-decoder*, cujos diferenciais são o módulo de agregação de contexto SPPM (Simple Pyramid Pooling Module) e o *decoder* FLD (Flexible and Lightweight Decoder) [Peng et al. 2022]. O SPPM é uma versão ajustada para execução em tempo real do *framework* PPM, que aplica três operações de *global-average pooling*, com *bin sizes* diferentes à saída do *encoder* antes de agregá-la ao *decoder* [Peng et al. 2022]. Já o FLD, diferente dos *decoders* convencionais, reduz gradualmente o número de canais durante o *upsample*, além de usar mecanismos de atenção, através de módulos chamados UAFM (Unified Attention Fusion Module), para fundir suas *features* com as *features* do *encoder* [Peng et al. 2022]. Para o *encoder*, escolhe-se uma rede neural "leve"como *backbone* (o artigo original fez testes usando a rede STDCNet) [Peng et al. 2022].

Para o treinamento do modelo, usaram-se dados do conjunto Clouds-1000, onde as anotações foram exportadas do Supervisely no formato PNG 8-bits. O *framework* utilizado para treino foi o PaddleSeg, dos mesmos criadores da rede PP-LiteSeg. A divisão dos dados em conjuntos de treinamento, validação e teste foi feita aleatoriamente pelo próprio PaddleSeg, nas proporções de 80%, 10% e 10% respectivamente. O treino foi realizado em um computador com Windows 10, equipado com um processador AMD Ryzen 7 1700X 3,4GHz, 16GB de RAM e uma placa de vídeo NVIDIA GeForce GTX 1070. Devido às limitações de configuração do dispositivo, o tempo total de treinamento foi de aproximadamente 3 dias.

As restrições de *hardware* tornaram bastante limitadas as escolhas de hiperparâmetros. Para o aprendizado ocorrer em tempo hábil, treinou-se o modelo 80.000 iterações, com *batch size* de 4 e *backbone* STDCNet. A resolução das imagens foi reduzida de 2592x1944 para 1296x962 para otimização. Aplicou-se *Data Augmentation* com inversão horizontal e variações de brilho, contraste e saturação em 25%. A escolha de funções de perda incluiu MultiClass Focal Loss e Dice Loss com pesos iguais. A *learning rate* foi $10^{-}4$, com *polynomial dacay* de $0,9$, utilizando o otimizador AdamW. Os parâmetros Beta1 e Beta2 foram configurados em $0,9$ e $0,999$, com decaimento de peso de $0,01$ para controlar a convergência do modelo.

3.3. Correção de Problemas de Classificação

Durante a análise qualitativa das classificações, observou-se que o modelo, ocasionalmente, cometia dois tipos de erros de segmentação. No primeiro deles, fragmentos de nuvens eram erroneamente classificados como um tipo diferente do restante da nuvem, como se houvesse uma nuvem dentro da outra. No segundo problema, uma mesma nuvem era classificada com diferentes tipos em uma sequência de imagens, como se a classe da nuvem pudesse mudar de minuto a minuto.

Para solucionar os problemas descritos acima, implementou-se um algoritmo de correção de problemas de classificação, que possui quatro etapas, conforme será explicado a seguir.

Na primeira etapa, removem-se as árvores e identificam-se as componentes conexas da imagem, de modo que cada componente corresponderá a uma nuvem. Na segunda etapa, para cada componente conexa, cria-se uma máscara com o formato da componente e aplica-se esta máscara às duas imagens imediatamente anteriores da sequência de imagens. Na terceira etapa, faz-se uma contagem dos *pixels* dentro da área da máscara, para

as três imagens (a imagem sendo tratada e as duas imagens anteriores), afim de descobrir a classe mais numerosa, que será considerada a classe predominante. Na quarta e última etapa, todos os *pixels* da componente conexa da imagem sendo tratada são reclassificados para a classe predominante.

Assim, resolvem-se os dois problemas: as nuvens fragmentadas passam a ter uma classe só e a classe é coerente com as imagens anteriores.

3.4. Rastreamento do Movimento

Conforme mencionado anteriormente, o método escolhido para o rastreamento do movimento de nuvens foi o fluxo óptico de Gunnar-Farneback, devido ao fato de ser um método de fluxo óptico denso e, portanto, ser mais vantajoso para analisar nuvens.

No fluxo óptico, busca-se o deslocamento de *pixels* entre uma imagem e outra em uma sequência de imagens, partindo da premissa que a intensidade dos *pixels* não se altera [Lin et al. 2023]. No fluxo óptico denso, o resultado do cálculo é o conjunto de vetores de deslocamento de cada pixel da imagem.

O método de Gunnar Farneback calcula os vetores de deslocamento entre duas imagens aproximando cada imagem a um polinômio quadrático. [Farnebäck 2003] Os coeficientes do polinômio da primeira imagem são calculados via uma versão ponderada do método dos mínimos quadrados, e os do segundo polinômio podem ser encontrados algebricamente [Farnebäck 2003].

Assim, busca-se calcular um deslocamento d, capaz de transformar o polinômio correspondente ao primeiro *frame*, dado por

$$f_1(x) = x^T A_1 x + b_1^T x + c_1$$

ao polinômio dado por

$$f_2(x) = f_1(x - d) = (x - d)^T A_1 (x - d) + b_1^T (x - d) + c_1$$

que corresponde ao *frame* subsequente [Farnebäck 2003].

Como é irreal representar os deslocamentos entre duas imagens somente com dois polinômios e um vetor, e como, na prática, o cálculo de d ponto a ponto produz muito ruído, o valor de deslocamento é estimado para a vizinhança de um ponto, com um erro mínimo [Farnebäck 2003].

Neste trabalho, o algoritmo de Gunnar Farneback é usado para encontrar os vetores de deslocamento dos *pixels* de uma nuvem. Esses vetores são usados para calcular um vetor predominante de movimento da nuvem. Este vetor predominante é obtido a partir da soma dos vetores dos *pixels* da nuvem divididos pelo número de *pixels*.

3.5. Geração de Vídeo

Os vídeos são gerados a partir de uma sequência temporal de imagens do horizonte, utilizando-se as bibliotecas OpenCV e FFMPEG do Python. Cada *frame* do vídeo corresponde a uma imagem da sequência. No vídeo, as nuvens aparecem segmentadas e com indicação do seu movimento.

A direção do vetor predominante de movimento das nuvens é representada no vídeo por setas, enquanto o tipo e a velocidade da nuvem são representado por um esquema de cores. Neste esquema de cores de nuvens, a luminosidade da cor da nuvem representa a magnitude do vetor de deslocamento, de modo que, quanto mais escura a nuvem, mais rápida ela é. Já a matiz da cor representa a classe a qual a nuvem pertence. A matiz de cor associada a cada classe de pixel do *dataset* pode ser visto na Tabela 2.

Tabela 2. Matiz de cor associada a cada classe no vídeo gerado.

Classe	**Matiz**
Estratocumuliforme	Ciano
Estratiforme	Magenta
Cirriforme	Amarelo
Cumuliforme	Vermelho
Árvore	Verde
Céu	Azul escuro

A geração do vídeo é feita a partir de *script* desenvolvido em Python, que recebe como entrada o diretório contendo a sequência de imagens de entrada, um diretório para armazenar as imagens geradas pelo segmentador semântico, o diretório onde o *framework* PaddleSeg está instalado, o caminho para o arquivo YAML com os pesos do modelo, o caminho completo para o arquivo de vídeo de saída, entre outros argumentos opcionais.

4. Resultados

A seguir, será feita análise quantitativa do desempenho do segmentador semântico a partir de métricas de avaliação de *machine learning* e serão detalhados casos onde há conhecida imprecisão na indicação do movimento das nuvens, além de problemas causados pelo algoritmo de correção de segmentação.

4.1. Desempenho do Modelo

A avaliação do modelo de segmentação semântica usado na solução foi feita logo após o seu treinamento, através do *script* **val.py**, fornecido pelo *framework* PaddleSeg. A Tabela 3 traz o resultado de métricas calculadas sobre os resultados.

Tabela 3. Resultados da avaliação do modelo.

Métrica	**Valor**
Intersecção sobre união (média)	0,6580
Acurácia	0,8698
Coeficiente Kappa	0,8245
F1-Score	0,7752

Observando a tabela, percebemos valores altos, acima de 0,8, para a acurácia e o Kappa. A acurácia alta indica que o modelo classifica os *pixels* corretamente na maioria dos casos, enquanto o Kappa alto indica que essa classificação correta não ocorre por acaso, ou seja, há uma boa correspondência entre predito e real.

Por outro lado, o F1-score e o IoU médio (Intersecção sobre União média) não atingiram valores tão elevados. Isto indica que o modelo, apesar de classificar bem *pixels* individuais, comete alguns erros ao segmentar a área das nuvens. O fato de o IoU médio ter um valor menor do que o F1-Score reforça essa hipótese, já que aquele penaliza mais fortemente imprecisões de segmentação do que este.

A Tabela 4 traz métricas aferidas por classe, o que traz possíveis explicações para o resultado de apenas 0,6 do IoU médio. Um dos fatores que diminui o IoU médio foram os resultados para a classe das nuvens cumuliformes, com IoU de apenas 0,3447.

Tabela 4. Resultados da avaliação do modelo (por classe).

	Céu	***Stratocumulus***	***Stratus***	***Cirrus***	***Cumulus***	**Árvore**
IoU	0,7826	0,7788	0,5047	0,5855	0,3447	0,9517
Precisão	0,8751	0,8585	0,6983	0,7392	0,8639	0,9689
Recall	0,8811	0,8935	0,6454	0,7379	0,3645	0,9817

O IoU baixo mencionado, em conjunto com o *Recall* baixo, de apenas 0,3645, indicam que o modelo está confundindo muitas nuvens cumuliformes com outras classes. Uma explicação plausível para isso seria o fato de que um percentual muito baixo, de apenas 9,04%, das imagens de treinamento possuem nuvens do tipo cumuliforme.

Por outro lado, as métricas para a classe estratocumuliforme, que aparece em 81,53% das imagens do *dataset*, de foram todas acima de 0,75. A classe céu, que corresponde aos *pixels* não anotados das imagens originais e, portanto, está virtualmente presente em todas as imagens, apresentou resultados parecidos com o da classe estratocumuliforme. Tudo isso nos indica que a presença da classe em um número significativo de imagens do *dataset* pode ser crucial para que se identifique mais nuvens desta classe corretamente.

É importante destacar que a classe Árvore teve desempenho excepcional, apresentando valor acima de 0,95 para todas as métricas. Isto se deve tanto ao fato de a classe estar presente em 99,3% dos dados de treinamento quanto ao fato destas serem imóveis e com posição fácil de predizer. Tais valores acima da média, aliados à grande quantidade de *pixels* que as árvores costumam ocupar em uma imagem, fazem subir o valor de métricas como o mIoU, acurácia, o F1-Score e o coeficiente Kappa.

4.2. Indicação Imprecisa da Direção de Movimento

Embora, no geral, a magnitude e direção do movimento das nuvens, calculada pela média de vetores de pixel obtidos via Gunnar-Farneback, corresponda ao movimento real da nuvem, e embora, na maioria dos casos, as cores e setas das nuvens representem corretamente, foram identificados três motivos pelos quais há situações onde as setas passam informações confusas ou imprecisas sobre as nuvens.

O primeiro motivo é a forma de representação escolhida para a indicação de movimento, onde mesmo uma nuvem parada ou semi-parada apresenta setas. Apesar de ser possível inferir, a partir da cor branca da nuvem, que seu movimento em determinada direção é insignificante, a indicação das setas pode dar a entender o contrário.

O segundo motivo decorre do fato de que o vetor de deslocamento das nuvens é uma média de vetores. Assim, em nuvens onde há *pixels* se movendo em várias direções, como, em nuvens que se expandem, contraem, ou sofrem algum tipo de distorção, as setas não necessariamente correspondem com o movimento real da nuvem, pois a média dos vetores (que provavelmente será baixa) pode não corresponder à transformação sofrida pela nuvem.

O terceiro motivo, que diferente dos demais, se aplica a praticamente todas as situações, é inerente aos métodos de fluxo óptico, pois estes partem da premissa de que a intensidade de *pixels* de uma imagem a outra é constante, o que dificilmente se verificar na realidade. Devido a este fator, é esperado certo grau de imprecisão no cálculo dos vetores de movimento em geral.

4.3. Reclassificação Indesejada no Pós-processamento

Há duas situações onde o algoritmo de pós-processamento, que corrige problemas de segmentação semântica, acaba, em vez disso, causando problemas.

Uma delas é quando há duas nuvens de classes diferentes sobrepostas ou encostadas uma na outra. Quando isto acontece, o algoritmo de correção, por detectar nuvens através de componentes conexas, acaba identificando as duas nuvens como uma só. A consequência disso é a reclassificação da nuvem menor para a classe da maior, pois a contagem de *pixels* identificará a classe da nuvem maior como sendo a predominante.

Outra problema ocorre quando uma nuvem em movimento ocupa espaço onde outras nuvens estavam em momentos anteriores. Em tal situação, a nuvem acaba sendo reclassificada para a classe das nuvens que antes ocupavam o espaço dela. A razão para isto é a etapa onde se projeta a máscara da nuvem nas duas imagens anteriores, pois esta projeção faz com que *pixels* das nuvens de classe diferente entrem para a contagem de *pixels*. Isto dá margem para a possibilidade que a classe dessas nuvens torne-se a predominante, forçando uma reclassificação indevida na nuvem original.

Decidiu-se manter o algoritmo de pós-processamento, apesar dos problemas identificados, devido à consideração de que é preferível errar ao afirmar que uma nuvem pertence à mesma classe que a do *frame* anterior do que cometer um erro ao afirmar que ela pertence a uma classe diferente. O segundo erro é considerado mais grave pois este compromete o rastreamento de movimento da nuvem, haja vista o funcionamento dos algoritmos de fluxo óptico, que detecta somente deslocamento de *pixels* de mesma intensidade, portanto, de mesma classe. Essa escolha visa otimizar o rastreamento de nuvens, priorizando-o em detrimento da classificação.

5. Conclusões e Recomendações para Trabalhos Futuros

Através da solução experimental implementada no presente trabalho, foi possível demonstrar a factibilidade de se efetuar classificação e rastreio de movimentação de nuvens em imagens do horizonte, haja vista o sucesso na criação de vídeos. A segmentação semântica, apresentou bons resultados, com acurácia de 0,8698, distinguindo eficientemente céu, nuvens e árvores, apresenta desafios ao lidar com classes menos representadas no conjunto de dados. Recomenda-se a adoção de medidas para aprimorar o desempenho, incluindo o balanceamento do conjunto de dados, a utilização de *hardware* mais avançado

para explorar hiperparâmetros de forma mais abrangente e a realização de comparações entre o PP-LiteSeg e outras técnicas de segmentação semântica.

Para cobrir as deficiências do modelo, foi necessário aplicar pós-processamento para corrigir eventuais erros de segmentação. Para evitar esta etapa corretiva, é crucial investir em uma maior quantidade e qualidade de dados. Isso inclui a aquisição de dados específicos para treinar o modelo a lidar com os problemas identificados, como a incorporação de imagens de nuvens próximas ou sobrepostas. Além disso, ajustes no rastreamento de nuvens são recomendados, como considerar a possibilidade de mudança de classe das nuvens ao longo do tempo. Essas medidas visam aperfeiçoar a segmentação desde a etapa inicial e reduzir a necessidade de intervenções pós-processamento.

O rastreamento de movimento das nuvens mostrou-se satisfatório, onde a representação do vetor predominante de movimento das nuvens, no geral, demonstra coerência com o deslocamento real. No entanto, há imprecisões conhecidas, que podem ser superadas adotando-se outra estratégia para lidar com elas. Abordagens possíveis seriam o cálculo de vetores de deslocamento por seção de nuvem em vez um único vetor, a adoção de outra forma de cálculo de vetor predominante que não média de vetores, e o uso diferentes técnicas de registro de imagens para rastrear o deslocamento de nuvens.

Referências

Arrais, J. M., Martins, B. J., Chaves, T. Z. L., Cerentini, A., Martins, S. L. M. N., and von Wangenheim, A. (2022). Systematic literature review on ground-based cloud tracking methods for nowcasting and short-term forecasting. Technical report, Brazilian Institute for Digital Convergence.

Barrett, E. C. and Grant, C. K. (1976). The identification of cloud types in landsat mss images. Technical report.

do Nascimento, L. R., de Souza Viana, T., Campos, R. A., and Rüther, R. (2019). Extreme solar overirradiance events: Occurrence and impacts on utility-scale photovoltaic power plants in brazil. *Solar Energy*, 186:370–381.

Farnebäck, G. (2003). Two-frame motion estimation based on polynomial expansion. In *SCIA13 : Gothenburg, Sweden*, number 2749 in Lecture Notes in Computer Science, pages 363–370.

Hill, C. A., Such, M. C., Chen, D., Gonzalez, J., and Grady, W. M. (2012). Battery energy storage for enabling integration of distributed solar power generation. *IEEE Transactions on smart grid*, 3(2):850–857.

Inman, R. H., Pedro, H. T. C., and Coimbra, C. F. M. (2013). Solar forecasting methods for renewable energy integration. *Progress in energy and combustion science*, 39(6):535–576.

Kumari, P. and Toshniwal, D. (2021). Deep learning models for solar irradiance forecasting: A comprehensive review. *Journal of Cleaner Production*, 318:128566.

Lin, F., Zhang, Y., and Wang, J. (2023). Recent advances in intra-hour solar forecasting: A review of ground-based sky image methods. *International Journal of Forecasting*, 39(1):244–265.

Martins, G. L., Neto, S. L. M., and Rüther, R. (2022). Evaluating the performance of radiometers for solar overirradiance events. *Solar Energy*, 231:47–56.

Peng, J., Liu, Y., Tang, S., Hao, Y., Chu, L., Chen, G., Wu, Z., Chen, Z., Yu, Z., Du, Y., Dang, Q., Lai, B., Liu, Q., Hu, X., Yu, D., and Ma, Y. (2022). Pp-liteseg: A superior real-time semantic segmentation model.

APÊNDICE B – *SCRIPT* CLOUDSEG.PY

```
import argparse
import os
from pathlib import Path
import cv2
import numpy as np

# argumentos

parser = argparse.ArgumentParser()
parser.add_argument("-d", "--dataset_dir", required=True, help="Dataset
↪  directory")
parser.add_argument("-i", "--inf_out_dir", required=True, \
    help="Inference output directory")
parser.add_argument("-s", "--run_segmentation", action='store_true',
↪  help="Run segmentation or just read inference folder annotations")
parser.add_argument("-p", "--ps_root_dir", required=True, help="PaddleSeg
↪  root path")
parser.add_argument("-m", "--model_yaml", required=True, help="PPLiteSeg
↪  model YAML path")
parser.add_argument("-v", "--video_file_path", required=True, help="Video
↪  output file path")
parser.add_argument("-f", "--video_frame_dur", default=3.0, help="Video
↪  frame duration (in seconds)")
parser.add_argument("-o", "--only_annotation", action='store_true',
↪  help="Video w/ annotation only or annotated original image")
parser.add_argument("-a", "--annotation_alpha", default=0.4,
↪  help="Annotation alpha: value from 0.0 (opaque) to 1.0 (full
↪  transparend) (for annotated original image only)")
parser.add_argument("-c", "--fourcc_string", default="mp4v", help="FourCC
↪  video codec code")
parser.add_argument("-g", "--generate_images", action="store_true",
↪  help="Generate image sequences instead of videos")
args = parser.parse_args()

dataset_dir = Path(args.dataset_dir)
inf_out_dir = Path(args.inf_out_dir)
run_segmentation = args.run_segmentation
```

```
infer_py = Path(args.ps_root_dir) / "deploy/python/infer.py"
model_yaml = Path(args.model_yaml)
video_file_path = Path(args.video_file_path)
frame_duration = float(args.video_frame_dur)
only_annotation = args.only_annotation
annotation_alpha = float(args.annotation_alpha)
annotation_alpha = float(args.annotation_alpha)
fourcc_string = args.fourcc_string
not_video = args.generate_images

# constantes

classes = {
    "CEU": 38,
    "STRATOCUMULUS": 75,
    "STRATUS": 113,
    "CUMULUS": 52,
    "CIRRUS": 14,
    "ARVORE": 89}

cor_classes = {
    classes["CEU"]: np.array([128, 64, 0]), # dark azure BGR
    classes["STRATOCUMULUS"]: np.array([180, 0.5, 1.]), # cyan HLS
    classes["STRATUS"]: np.array([300, 0.5, 1.]), # magenta HLS
    classes["CUMULUS"]: np.array([45, 0.5, 1.]), # gold HLS
    classes["CIRRUS"]: np.array([0., 0.5, 1.]), # red HLS
    classes["ARVORE"]: np.array([0, 192, 0])} # green BGR

# funcoes

def obter_componentes_conexas(img):
    img_ceu_nuvem = np.array(img, copy=True)

    # remove ceu e arvore
    img_ceu_nuvem[img_ceu_nuvem == classes["CEU"]] = 0
    img_ceu_nuvem[img_ceu_nuvem == classes["ARVORE"]] = 0

    img_ceu_nuvem[img_ceu_nuvem != 0] = 255 # qualquer nuvem = 255
```

```
    return cv2.connectedComponents(img_ceu_nuvem, 8, cv2.CV_32S) # obtem
    ↪ componentes conexas

def calcular_tipo_predominante(actual, prev, prev2, num_comps,
↪ comps_actual):
    for num_componente in range (1, num_comps): # para cada componente
    ↪ conexa da imagem...
        pxs_componente = np.where(comps_actual == num_componente)  #
        ↪ obtem pixels da componente

        pxs_por_tipo = {}
        for img in [actual, prev, prev2]: # para imagem atual e 2
        ↪ anteriores...
            (tipos, quantidades) = np.unique(img[pxs_componente],
            ↪ return_counts=True) # conta tipos da componente

            # soma cada tipo encontrado na lista de tipos das 3 imagens
            for tipo, quantidade in zip(tipos, quantidades):
                if tipo != classes["CEU"] and tipo != classes["ARVORE"]:
                ↪ # desconsidera o que não for nuvem
                    pxs_por_tipo[tipo] = pxs_por_tipo[tipo]+quantidade if
                    ↪ tipo in pxs_por_tipo else quantidade

        tipo_pred = max(pxs_por_tipo, key=pxs_por_tipo.get) # obtem tipo
        ↪ mais encontrado nas 3 imagens
        actual[pxs_componente] = tipo_pred # pinta parte correspondente a
        ↪ componente na imagem com o tipo predominante

def vetor_predominante_componente(fluxo, num_comp, comps):
    # pega indices só dos pixels da nuvem de interesse
    idx_px_nuvens = np.where(comps == num_comp)

    # corrige y
    idx_px_nuvens_y = idx_px_nuvens[0]
    idx_px_nuvens_y = idx_px_nuvens_y[np.where((idx_px_nuvens_y >= 0) |
    ↪ (idx_px_nuvens_y < img_height))]
    # corrige x
    idx_px_nuvens_x = idx_px_nuvens[1]
```

```
    idx_px_nuvens_x = idx_px_nuvens_x[np.where((idx_px_nuvens_x >= 0) |
    ↪  (idx_px_nuvens_x < img_width))]

    # obtem vetor médio dentre vetores dos pixels da nuvem
    vetor_y = fluxo[idx_px_nuvens_y, idx_px_nuvens_x, 1].mean()
    vetor_x = fluxo[idx_px_nuvens_y, idx_px_nuvens_x, 0].mean()

    # retorna vetor em coordenadas polares
    mag, ang = cv2.cartToPolar(vetor_x.item(), vetor_y.item())
    return [mag[0,0], ang[0,0]]

def vetores_predominantes_img(img_antes, img_depois, num_comps,
↪  comps_antes):
    fluxo = cv2.calcOpticalFlowFarneback(img_antes, img_depois, None, 0.5,
    ↪  3, 15, 3, 5, 1.2, 0)
    return np.array([vetor_predominante_componente(fluxo, num_comp,
    ↪  comps_antes) for num_comp in range(1, num_comps)])

def obter_pontos_seta(tam_seta, esp_seta, angulo):
    # obtem ponto final da seta
    x, y = cv2.polarToCart(tam_seta - esp_seta, angulo)
    end = np.array([round(x[0, 0]), round(y[0, 0])])

    # desloca pontos inicial e final para reta passar pelo ponto central
    ↪  da imagem
    delta_x = (tam_seta - end[0])/2
    delta_y = (tam_seta - end[1])/2
    start = np.array([delta_x, delta_y])
    end = end + start

    return start, end

def desenhar_seta(img, x_seta, y_seta, tam_seta, esp_seta, angulo, cor):
    # obtem pontos da seta
    start, end = obter_pontos_seta(tam_seta, esp_seta, angulo)

    # desloca pontos para local da imagem onde seta deve ser desenhada
    desloc = np.array([x_seta, y_seta])
    start = start + desloc
```

```
    end = end + desloc

    # desenha seta e retorna
    return cv2.arrowedLine(img, start.astype(int), end.astype(int),
    ↪  tuple(cor.tolist()), esp_seta, tipLength = 0.5)

def criar_img_setas(angulo, cor_seta, cor_fundo):
    img = np.ones((img_height, img_width, 3)) * cor_fundo

    tam_seta = 25
    esp_seta = 3

    for y in range(0, img_height, tam_seta):
        for x in range(0, img_width, tam_seta):
            desenhar_seta(img, x, y, tam_seta, esp_seta, angulo,
            ↪  cor_seta)

    return img

def mag_to_light(mag, max_mag):
    norm = cv2.normalize(np.array([0, mag, max_mag]), None, 0, 0.7,
    ↪  cv2.NORM_MINMAX)
    return 1. - norm[1]

def hsl_arr_to_bgr(hsl_arr):
    bgr = cv2.cvtColor(np.array([[hsl_arr]], dtype=np.float32),
    ↪  cv2.COLOR_HLS2BGR)[0,0]
    bgr = np.array(bgr*255, dtype=np.uint8)
    return bgr

def calcula_cores_fundo_seta(cor_tipo, magnitude):
    cor_mag = cor_tipo
    cor_seta = np.array(cor_mag, copy=True)

    # calcula cor do fundo e da seta
    cor_mag[1] = mag_to_light(magnitude, max_mag) # luminosidade de
    ↪  acordo com magnitude
    cor_seta[1] = cor_mag[1] - 0.3 # seta é sempre mais escura
```

```
    # converte pra BGR pra poder desenhar
    cor_mag = hsl_arr_to_bgr(cor_mag)
    cor_seta = hsl_arr_to_bgr(cor_seta)

    return cor_mag, cor_seta

def obter_img_bgr(img, num_componentes, componentes_img, vetores_img,
↪  max_mag):
    img_base = np.zeros((img_height, img_width, 3), dtype=np.uint8) #
    ↪  futura imagem com nuvnes/setas

    # para cada nuvem (componente conexa) da imagem...
    for num_comp in range(1, num_componentes):
        pxs_componente = np.where(componentes_img == num_comp) # pixels
        ↪  da nuvem
        cor_tipo =  np.array(cor_classes[img[pxs_componente][0]],
        ↪  copy=True) # obtem cor a partir do tipo

        if vetores_img is None:
            # se nao houver vetor, pinta imagem sem seta
            pintura = hsl_arr_to_bgr(cor_tipo)
        else:
            mag, ang = tuple(vetores_img[num_comp-1])

            # se houver vetor, pinta imagem com seta
            cor_fundo, cor_seta = calcula_cores_fundo_seta(cor_tipo, mag)
            setas = criar_img_setas(ang, cor_seta, cor_fundo)
            pintura = setas[pxs_componente]

        img_base[pxs_componente] = pintura # pinta pixels de interesse

    # pinta pixels de ceu e arvore
    for tipo in ["CEU", "ARVORE"]:
        img_base[np.where(img == classes[tipo])] =
        ↪  cor_classes[classes[tipo]]

    return img_base

# Segmenta nuvens com PaddleSeg
```

```
if run_segmentation:
    os.system(f"python {infer_py} \
          --config {model_yaml} \
          --image_path {dataset_dir} \
          --save_dir {inf_out_dir}")

# Obtem np arrays das imagens anotadas
print("Reading images... ", end=" ")
imgs = [cv2.imread(str(img), cv2.IMREAD_GRAYSCALE) for img in
↪  Path(inf_out_dir).glob('*.*')]
img_height, img_width = np.shape(imgs[0])
print("OK")

# Extrai componentes conexas das imagens anotadas
comps_imgs = [obter_componentes_conexas(img) for img in imgs]

# Calcula tipo predominante da ultima imagem pra cada trio de imagens
print("Calculating predominant cloud type... ", end=" ")
for img1, img2, img3, comps3 in zip(imgs, imgs[1:], imgs[2:],
↪  comps_imgs[2:]):
    calcular_tipo_predominante(img3, img2, img1, *comps3)
print("OK")

# Calcula vetores predominantes
print("Calculating cloud displacement vectors... ", end=" ")
vetores_imgs = [vetores_predominantes_img(i1, i2, *c1) for i1, i2, c1 in
↪  zip(imgs, imgs[1:], comps_imgs)]
max_mag = max([v[..., 0].max() for v in vetores_imgs]) # magnitude
↪  maxima, p/normalizacao
print("OK")

# Gera imagem colorida com setas para cada imagem
print("Generating video frame images... ", end=" ")
vetores_imgs.append(None)
color_imgs = np.array([obter_img_bgr(img, *cs, vs, max_mag) for img, cs,
↪  vs in zip(imgs, comps_imgs, vetores_imgs)])

# Mescla foto com anotacoes com seta geradas, se for o caso
if not only_annotation:
```

```
    # lê fotos como BGR
    orig_imgs = np.array([cv2.imread(str(img)) for img in
    ↪  Path(dataset_dir).glob('*.*')])

    # redimensiona anotacoes para o tamanho das fotos
    img_height, img_width = np.shape(orig_imgs)[1:3]
    color_imgs = np.array([cv2.resize(x, (img_width, img_height)) for x
    ↪  in color_imgs], dtype=np.uint8)

    # remove anotacao do ceu (sustitui por pixels de foto do ceu)
    pxs_ceu = np.where(np.all(color_imgs == cor_classes[classes["CEU"]],
    ↪  axis=-1))
    color_imgs[pxs_ceu] = orig_imgs[pxs_ceu]

    # mescla fotos com anotacoes
    beta = 1 - annotation_alpha
    color_imgs = [cv2.addWeighted(o, annotation_alpha, c, beta, 0.0) for
    ↪  o, c in zip(orig_imgs, color_imgs)]
print("OK")

# Gera video
print("Generating video... ", end=" ")
frames = len(imgs)
if not_video:
    for f in range(frames):
        cv2.imwrite(str(video_file_path / f"frame{f}.png"),
        ↪  color_imgs[f])
else:
    fps = 30
    fourcc = cv2.VideoWriter_fourcc(*fourcc_string)
    video = cv2.VideoWriter(str(video_file_path), fourcc, fps, (img_width,
    ↪  img_height))
    for f in range(frames):
        for i in range(int(fps * frame_duration)):
            video.write(color_imgs[f])
    video.release()
print("OK")
print("Finished!")
```