Talhonamento em sensoriamento remoto agrícola: Abordagens utilizando grafos

Mateus Pinto da Silva[1]

[1]Ciência da Computação – Universidade Federal de Viçosa (UFV-caf) – Florestal – MG – Brasil

(mateus.p.silva[1])@ufv.br

Resumo. *Em sensoriamento remoto, segmentar talhões agrícolas, aplicação conhecida como talhonamento. Nesse sentido, foram adaptados e analisados dois algoritmos de grafos para o processo: Breadth-First Search baseada em Busca Local e o algoritmo de Felzenszwalb baseado em Minimum Spanning Tree para tal. Os modelos foram testados no Dataset PASTIS, que usa séries temporais de imagens de talhões agrícolas da França capturadas pelo satélite Sentinel-2. Os resultados parecem qualitativamente promissores, e a tarefa de utilizar o conteúdo visto na disciplina INF610 em algo relacionado a pesquisa foi concluída.*

1. Introdução

Em sensoriamento remoto, segmentar talhões agrícolas, aplicação conhecida como talhonamento. Nela, dada uma ou mais imagens de satélite, os talhões agrícolas são segmentados, tendo como resultado ou uma nova imagem em que cada talhão recebe um diferente indicador (normalmente explicitado por uma cor diferente nas visualizações), ou uma lista de várias geometrias, cada uma sendo um talhão. Grafos são a estrutura mais genérica de Ciência da Computação, e podem e foram utilizados para representar imagens. Assim, é possível converter o problema de talhonamento para o de divisão de grafos.

Já existem algumas metodologias de segmentação de imagens utilizando grafos, entretanto sensoriamento remoto tem suas peculiaridades. Os satélites, em geral, são multiespectrais ou hiperespectrais (ou seja, apresentam um número maior do que três bandas). Nuvens e sombras de nuvens são bastante problemáticas, visto que impedem ou atrapalham bastante a visualização da superfície da Terra, e consequentemente reduzem a informação presente em cada imagem. Além disso, o uso de séries temporais de imagens são cruciais na agricultura, visto que a diferenciação de certos talhões são apenas vistos em certas imagens, pois é necessário que as plantas estejam em etapas diferentes de crescimento ou ainda que estejam tão crescidas a ponto de suas espécies serem diferenciáveis.

Como objetivos, destacam-se a criação de adaptações de metodologias de grafos para talhonamento, e também a conclusão do Projeto Final da disciplina de INF610. Nesse sentido, este trabalho encaixa-se na primeira opção disponibilizada pelo professor, que é a de: “Adaptação de alguma ED não trivial“.

Foram implementadas e adaptadas 2 metodologias de grafos para talhonamento, que são baseados conceitos diferentes de grafos, sendo elas: Breadth-First Search baseada em Busca Local e o algoritmo de Felzenszwalb baseado em Minimum Spanning Tree. A continuação do relatório está dividida em: a seção 2 descreve o dataset utilizado, a seção

3 aborda as técnicas de pré-processamento dos dados, a seção 4 detalha a construção dos grafos, a seção 5 apresenta as metodologias de segmentação desses grafos, 6 discute os resultados e, por fim, 7 traz as considerações finais do projeto.

2. Os dados usados

Para o desenvolvimento do trabalho, foram utilizados dados do satélite Sentinel-2, que é um satélite feito para monitoramento de vegetação e com dados públicos. O Dataset escolhido foi o PASTIS, que é um dataset de benchmark para segmentação semântica e panóptica de talhões agrícolas da França [Sainte Fare Garnot and Landrieu 2021]. De forma técnica, tal Dataset conta com várias instâncias de talhonamento do tamanho de 128x128, contendo um tensor com identificador único para cada um dos talhões. Além disso, conta com uma série temporal de tamanho [T, 10, 128, 128], sendo T uma quantidade variada de imagens por instância de talhonamento, e 10 a quantidade de bandas do satélite Sentinel-2 (Veja a Figura 2).

Infelizmente, as séries temporais não vêm com tratamento de nuvens (ou seja, remoção delas) nem mesmo com nenhum tipo de banda de qualidade (que informa quais pixels são nuvens ou sombras de nuvem). Assim, pré-processamento é necessário.

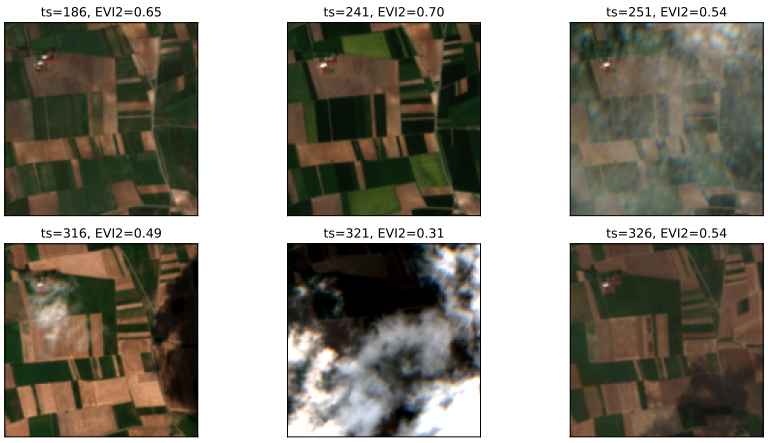


Figura 1. Alguns dados de uma série temporal do Dataset PASTIS, RGB.

3. Pré-processamento dos dados

Portanto, foi realizado uma série de pós-processamento sobre os dados, a fim de reduzir/eliminar nuvens e sombra de nuvens. O satélite Sentinel-2 é popularmente conhecido por seu problema com elas, visto que ele não apresenta banda térmica, o que dificulta muito a detecção especialmente de sombras de nuvem. Primeiro as imagens tiveram seus valores escalados para de 0 a 1, e adicionado um ganho de constraste, multiplicando todas as imagens por Pi. Depois, foi feito um corte seco das nuvens através da banda azul, ou seja, pixels cuja banda azul maior que 0.5 foram retirados. O mesmo foi feito com pixels com banda NIR menor que 0.3, a fim de retirar sombras de nuvem. Tal técnica é bastante comum e simples para tal filtro em agricultura, visto que nada no meio agrícola apresenta reflectância alta azul além das nuvens, e nada apresenta reflectância NIR muito baixa além das sombras.

Depois disso, as imagens com mais de 70% dos pixels retirados foram totalmente excluídas das séries. A primeira imagem teve seus pixels excluídos preenchidos pela média de cada banda, assim resultando numa imagem novamente totalmente preenchida. A seguir, os pixels nulos da segunda imagem foram preenchidas com a primeira, e assim sucessivamente. Foi aplicado, pixel a pixel na série toda, a suavização Savitzky-Golay com tamanho de janela 11 e ordem do polínomio igual a 3. Por fim, foi adicionado o Enhanced Vegetation Index 2 (que é uma combinação entre as bandas Vermelho e NIR), e os valores foram todos limitados entre 0 e 1. Mesmo assim, as nuvens ainda deixam resquícios no Dataset, mas de forma bem menor (veja a Figura 3).

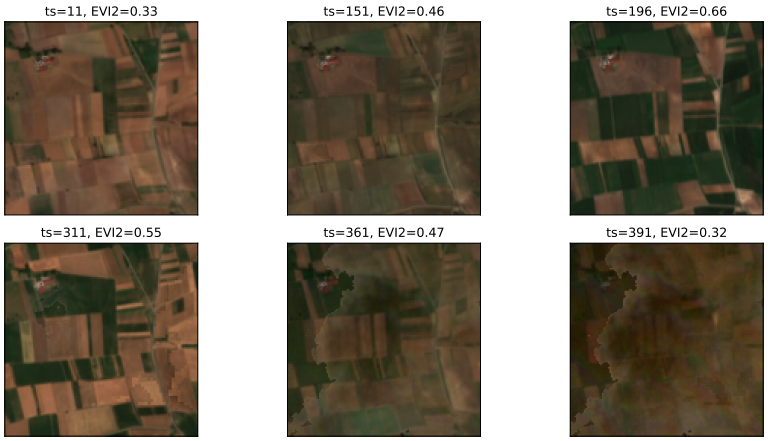


Figura 2. Dados tratados com nuvens removidas, RGB.

4. Construção dos grafos

Para a construção dos grafos, como é bem comum em grafos de imagens, cada pixel representa um vértice do grafo, e as arestas são as ligações entre os pixels, ou seja, cada vértice pode ter de 2 arestas (os pixels dos cantos da imagem) até quatro vértice, pois representam se um pixel está ao lado ou acima ou abaixo de outro.

Cada série temporal foi convertida num único gráfico (veja a figura 4), e cada série de pixels foi convertida em features específicas para a agricultura. Elas são: tamanho da senescência (ou seja, quantidade de dias entre a senescência e a colheita da planta), ganho de EVI2 (diferença entre EVI2 inicial e máximo, dividido pela quantidade de dias até o pico vegetativo), perda de EVI2 (diferença entre EVI2 máximo e final, dividido pela quantidade de dias do pico vegetativo até o dia final), EVI2 máximo, EVI2 mínimo e a cor do pico vegativo, i.e., como estava cada uma das bandas do Sentinel-2 no dia do pico vegetativo. Além disso, todos os valores são modularizados. Cada aresta mantém a distância euclideana das Features de um vértice para outro.

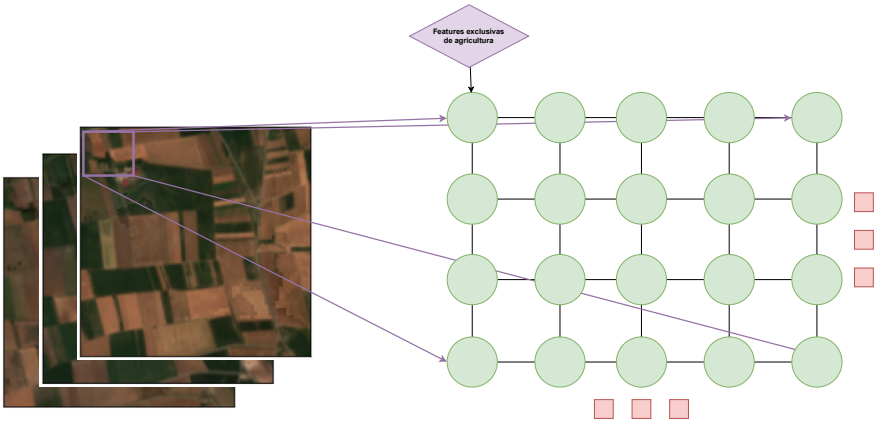


Figura 3. Grafo construído a partir de série temporal.

5. Metologias de segmentação

Foram construídas 2 metodologias de segmentação, baseadas em abordagens diferentes de grafos. Todas utilizam o grafo feito na seção anterior, e apresentam algumas diferenças de aplicação sendo elas as adaptações de cada algoritmo. Quanto a implementação, foi utilizada a representação em tensor das features por vértice, visto que são mais performáticos em memória e se adaptam bem a visão computacional em grafos.

5.1. Breadth-First Search

A primeira metodologia trata-se da Breadth-First Search, ou Busca em Largura, que é uma Busca Local de grafo. A ideia por de trás dela é bastante intuitiva: caso se escolha um pixel aleatório, muito provavelmente este pixel estará dentro de um talhão. Assim, basta fazer uma busca em largura escolhendo quais pixels provavelmente ainda são do mesmo talhão, e assignar um valor único para ele. Quando um pixel for diferente demais do pixel anterior, aquilo será considerado uma borda. O processo se repete até que não existam mais pixels que não foram visitados, ou seja, são usadas várias vezes a Busca Local. Além disso, o talhão só é considerado se tiver pelo menos uma determinada área, medida em pixels. Isso se dá pois há um valor de área mínimo para talhões agrícolas. O pseudocódigo segue:

```
def bfs_talhonamento(grafo, limiar_distancia, min_area_talhao):
    def retorna_vizinhos(i, j):
        movimentos = [(0, 1), (0, -1), (1, 0), (-1, 0)]
        movimentos = [(i + x, j + y) para cada (x, y) em movimentos]
        movimentos = filtra e mantem aqueles que estao dentro dos
    limites do grafo
        return movimentos

    visitados = mascara de booleanos falsos do tamanho de vertices X
    vertices
    mascara = mascara de inteiros zeros do tamanho de vertices X
    verticesfo

    def bfs_de_um_ponto(x, y, id, mascara):
        pilha = adiciona (x, y) a lista
        visitados[x, y] = Verdadeiro
```

```
14          sub_mascara = grafo de inteiros zeros do tamanho de grafo
15
16          while pilha nao estiver vazia:
17              x, y = pilha remove o ultimo elemento
18
19              for (i, j) em retorna_vizinhos(x, y):
20                  if nao visitados[i, j]:
21                      if (dist_eucl((x, y), (i, j)) < limiar_distancia):
22                          sub_mascara[i, j] = id
23                          pilha.adiciona(i, j)
24                      visitados[i, j] = Verdadeiro
25
26          if numero de pixels em sub_mascara > min_area_talhao:
27              mascara += sub_mascara
28              passos.adicionar(mascara.copia())
29
30      id = 1
31      while existem pontos nao visitados:
32          indice_x, indice_y = obtenha indice aleatorio nao visitado
33          bfs_de_um_ponto(indice_x, indice_y, id, mascara)
34          id += 1
35
36      return mascara
```

Ademais, é feita uma cópia da máscara a cada talhão adicionado, para que seja possível visualizar o processo com animações. Quanto a complexidade, é bastante difícil estimar usando métodos matemáticos a complexidade algorítmica devido ao loop *while*. Além disso, não parece muito trivial deduzir uma versão recursiva do algoritmo. Definindo a operação elementar de grafos como a expansão do nó (o que é bastante comum para algoritmos de grafos), e considerando o loop definido na linha 36 do pseudocódigo, é possível mostrar que todos os vértices são expandidos. Assumindo V como o número de vértices, a complexidade será para todos os casos:

$$T(V) = O(V)$$

Fiz também uma versão sem a retirada dos talhões pequenos (que são ruídos). Entretanto, o resultado só é pior, e a complexidade é a mesma.

5.2. Minimum spanning tree - Felzenszwalb

A próxima metodologia que adotei foi uma adaptação da segmentação de grafos de [Felzenszwalb and Huttenlocher 2004], porém híbrida usando Aprendizado de Máquina Não-Supervisionado. A técnica original funciona a partir da construção de Minimum Spanning Trees, e parece um pouco com a minha implementação anterior, porém usando uma abordagem gulosa. Todas as arestas são clusterizadas a partir de um Kmeans sobre dois clusters, a fim de separar quais arestas são as que unem talhões, e quais arestas são as separações entre os talhões. Como o Kmeans é não supervisionado, é feita uma análise da média de distância euclideana de cada cluster, e o com maior distância média é selecionado. Todas as arestas de uma cópia do grafo são removidas. As arestas do cluster selecionado são ordenadas da com menor peso pra maior, e elas são gradualmente adicionadas ao novo grafo, resultando em várias MSTs. De forma experimental, um bom valor de adição é o de 70% das arestas desse cluster emitido pelo KMeans. Ao fim, vários candidatos a talhões são MSTs do grafo original, e as muito pequenas são removidas, visto

que talhões agrícolas muito pequenos não existem. O pseudocódigo segue:

```
1  def felzenszwalb_talhonamento(grafo, percentil_corte=0.7,
       area_minima_talhao=40):
2      escalonador = StandardScaler()
3      kmeans = KMeans(n_clusters=2)
4
5      arestas = retorna_todas_arestas(grafo)
6      arestas_escalonadas = escalonador.escalar(arestas[:, 5:])
7      kmeans.clusterizar(arestas_escalonadas)
8
9      marks = kmeans.predict(arestas_escalonadas)
10     primeiro_cluster = arestas[marks == 0]
11     segundo_cluster = arestas[marks == 1]
12
13     if media(primeiro_cluster) < media(segundo_cluster):
14         primeiro_cluster, segundo_cluster = segundo_cluster,
       primeiro_cluster
15
16     novo_grafo = grafo.copiaSemArestas()
17
18     for aresta in segundo_cluster.ordenado().corte(percentil_corte):
19         novo_grafo.adicionaAresta(aresta)
20
21     for mst in novo_grafo.listaMSTS():
22         if mst.tamanho() < 40:
23             novo_grafo.removeMST(mst)
24
25
26     return novo_grafo
```

Novamente, a análise não é muito trivial, visto que o limite do loop importante que é o da linha 13 é desconhecido, visto que depende da clusterização do K-means. O K-means em si, no caso médio, é conhecido por ser linear [Arthur and Vassilvitskii 2006] a própria entrada. As entradas aqui são todas as arestas do grafo. Analisando o pior caso, em que todas arestas menos uma seriam definidas num dos clusters, assumindo a operação de adicionar aresta como operação elementar. Assim, seriam executadas $\widehat{0.7 \times A}$ vezes (sendo A o número de arestas) o que também é linear. Partindo do número de vértices, é possível calcular o número de arestas pois o grafo é sempre matricial. Assim, a complexidade será (assumindo o primeiro O(A) como a complexidade do K-Means):

$$T(A) = O(A) + \widehat{0.7 \times A}$$

$$V = (A-1) \times (A) + (A-1) \times (A)$$

$$T(V) = O(V) + O(V)$$

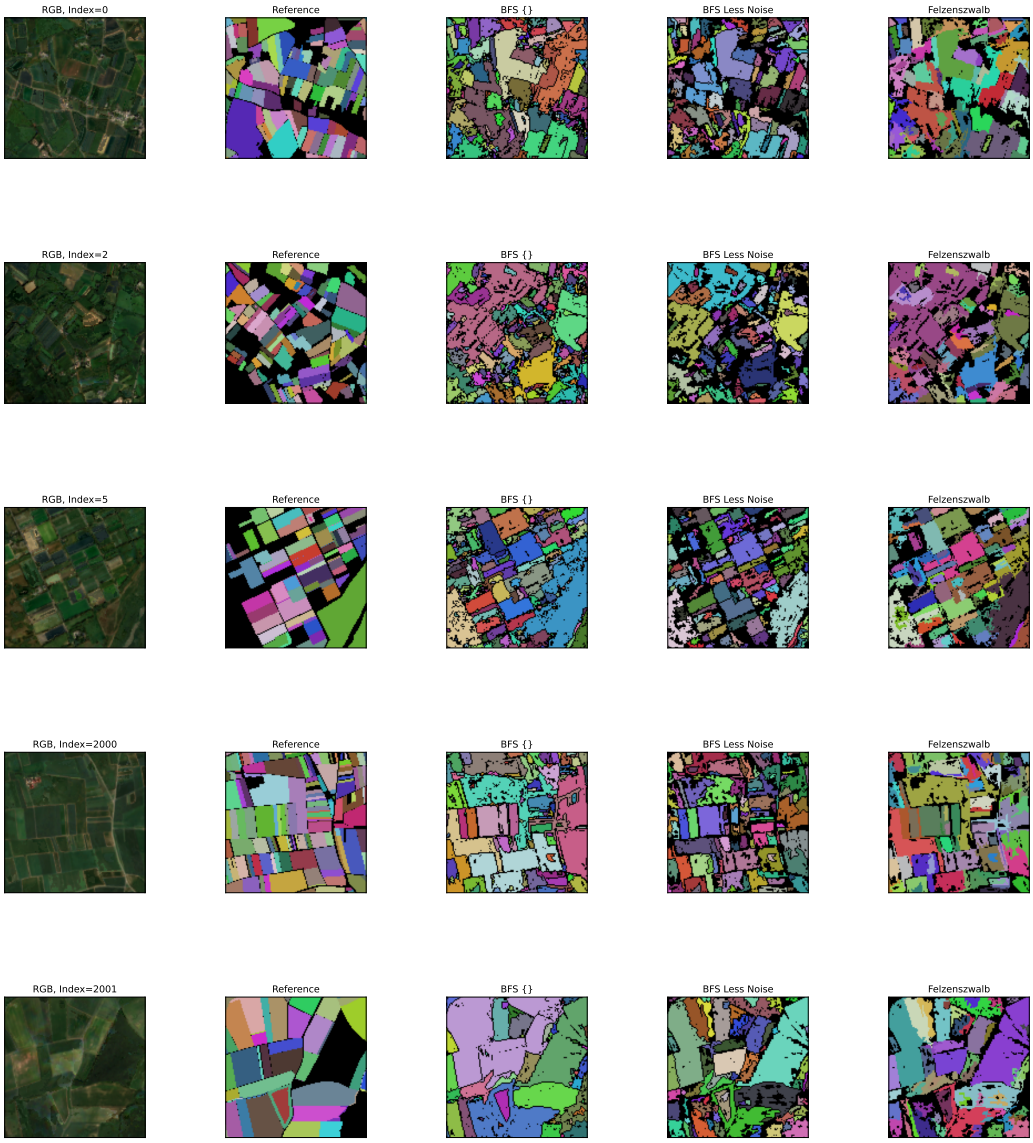
$$T(V) = O(V)$$

A complexidade no pior caso será não-polinomial, porém isso se deve a complexidade do K-Means.

6. Resultados

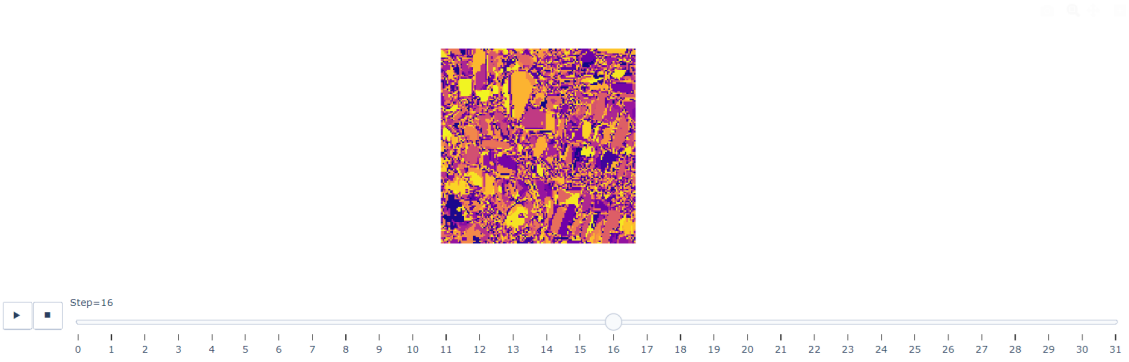
Como resultados, foi feita uma analise qualitativa. É importante notar que, como normalmente acontece nos algoritmos de visão computacional tradicionais, a qualidade dos

daqui desenvolvidos dependem bastante dos hiperparâmetros. Assim, serão mostrados alguns casos que os algoritmos funcionaram bem, lado a lado com um mosaico RGB do pico vegetativo por pixel.



Qualitativamente, quando alinhado com bons valores de hiperparâmetros, os algoritmos desenvolvidos parecem apresentar resultados interessantes. Porém, seria interessante forçar uma supersegmentação, i.e., dividir os talhões para mais do que é visto, para depois classificá-los utilizando algum modelo de classificação de série temporal. Assim, talhões lado-a-lado com a mesma cultura (e talvez mesma data de emergência e colheita) poderiam ser novamente unidos, gerando uma segmentação mais precisa. Esse tipo de pipeline é bastante usual no mercado e na literatura.

Além disso, é **IMPORTANTE** salientar ao leitor que foi desenvolvido uma visualização com animação interativa no código entregado junto deste relatório, que executa todos os algoritmos passo a passo. A visualização do funcionamento acontece bem melhor lá.



7. Considerações finais

Foram implementadas 2 metodologias diferentes, ambas adaptadas ao contexto de talhonamento, utilizando grafos, sendo elas a: Breadth-First Search baseada em Busca Local e o algoritmo de Felzenszwalb baseado em Minimum Spanning Tree. Embora as metodologias sejam bastante distantes da qualidade de segmentação estado da arte (que em geral, utiliza modelos de Deep Learning), creio que o objetivo do trabalho de adaptar algoritmos visto na disciplina para o contexto da pesquisa foi atingido. Além disso, ambas metodologias contaram com ajustes interessantes para lidar com séries temporais de imagens multiespectrais que são comuns em talhonamento agrícola. Além disso, o uso de K-Means na segunda metodologia reduziu a quantidade de pixels ruidosos no talhonamento.

Quanto a melhorias, é possível pensar em filtros que preencham os buracos que são ruídos nos talhões, além de mudar os hiperparâmetros para supersegmentação somado ao uso de modelos de classificação para, em casos de talhões com a mesma classe um ao lado do outro, os unir. Assim, o resultado melhoraria consideravelmente.

Referências

[Arthur and Vassilvitskii 2006] Arthur, D. and Vassilvitskii, S. (2006). How slow is the k-means method? In *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*, SCG '06, page 144–153, New York, NY, USA. Association for Computing Machinery.

[Felzenszwalb and Huttenlocher 2004] Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International journal of computer vision*, 59:167–181.

[Sainte Fare Garnot and Landrieu 2021] Sainte Fare Garnot, V. and Landrieu, L. (2021). Panoptic segmentation of satellite image time series with convolutional temporal attention networks. *ICCV*.