Construction parallèle d'arbre binaire de partition (BPT) pour la segmentation hiérarchique d'images

Mots-clés: structure hiérarchique, segmentation hiérarchique d'images, programmation parallèle, performance.

Contexte

Un arbre binaire de partition ou BPT (*Binary partition tree*) [1] est une représentation hiérarchique d'une image. Plus précisément, c'est un arbre binaire dont chaque nœud correspond à un ensemble de régions voisines fusionnées avec une certaine cohérence. Un nœud peut être soit une feuille représentant une région *élémentaire* ou bien l'union de deux régions voisines. La racine de l'arbre, quant à elle, correspond au support de l'image. Une telle structure peut être interprétée comme un ensemble de régions contenues dans une structure d'arbre hiérarchique. Les plus grandes régions sont proches de la racine tandis que les petites régions détaillées se trouvent près des feuilles.
Le BPT a démontré son utilité dans plusieurs domaines impliquant la détection d'objets [2]. Entre autres, le BPT est assez populaire dans le domaine de la télédétection [3, 4] où les objets d'intérêts dans les images peuvent être à la fois hétérogènes et complexes (e.g. quartier avec des maisons détaillées vs. lac homogène au milieu de la ville). Le BPT est une structure de données pertinente pour concevoir des outils d'analyse et de traitement d'images (e.g. analyse de vidéo, imagerie médicale). Une coupe ou *cut* [5] effectuée sur le BPT, comme illustrée par la figure 1a, permet d'obtenir une partition correspondant à une segmentation de la scène de l'image à un niveau d'échelle donné.

(a) (b)

Figure 1: Illustration, (a) de gauche à droite, d'une image de télédétection, d'une coupe effectuée sur un BPT, d'une partition correspondant à une segmentation de l'image à l'issue de la précédente coupe et enfin d'une mise en évidence des bordures des régions obtenues. (b) Création ascendante d'un BPT en partant des feuilles jusqu'à la racine.

Plusieurs coupes effectuées sur le BPT, comme illustrées par la figure 2, permettent donc d'obtenir plusieurs partitions qui correspondent à plusieurs niveaux de segmentation de l'image où l'on peut voir, suivant l'échelle choisie, les différents détails des différentes régions.
La construction d'un BPT se fait de façon ascendante (c.f. 1b) en partant par la détermination des feuilles – fournies par une partition initiale de l'image – vers l'obtention de la racine. Cela se fait en choisissant et en fusionnant itérativement deux régions adjacentes qui minimisent un critère de similarité. Les régions les plus proches sont donc fusionnées. Cette séquence de fusion est stockée dans le BPT qui modélise l'image à différentes échelles. Contrairement aux autres modèles hiérarchiques comme le max-tree, le min-tree, l'arbre des composants [6] et l'arbre de formes [7] dont les constructions se basent uniquement sur des informations intrinsèques de l'images, le BPT requiert en plus des informations externes liées aux connaissances d'un expert sur la scène observée (e.g. médecin, géographe). À partir d'une image I, une partition initiale Ω est déterminée (e.g. tous les pixels de l'image, zones plates ou superpixels [8, 9]), un grand nombre de BPTs différents peuvent être obtenus. Il est donc primordial de bien choisir l'ordre de fusion entre deux nœuds / régions adjacentes $N_i, N_j \in \mathcal{P}$ où $\mathcal{P}$ est une partition quelconque de Ω. Un tel choix nécessite la définition d'un modèle de région $M_r(N_i)$ qui spécifie comment une région N_i est caractérisée (e.g. couleur, forme, texture); et d'un critètre de fusion $O_r(N_i, N_j)$ qui défini une métrique qui permet

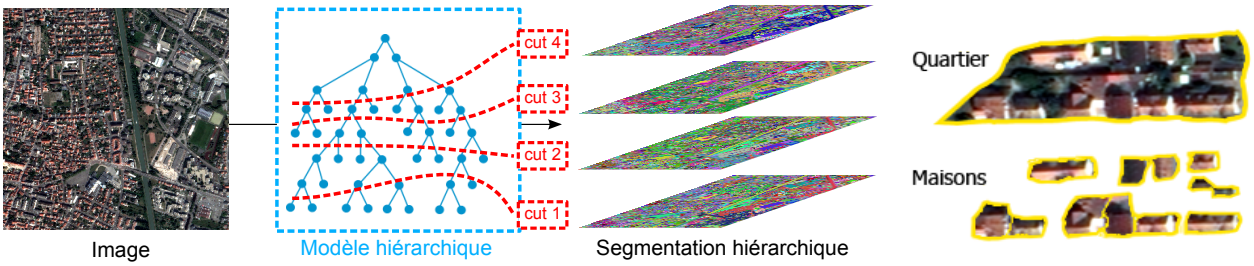

Figure 2: Illustration, de gauche à droite, d'une image de télédétection, de plusieurs coupes effectuées sur plusieurs niveaux d'un modèle hiérarchique (ici BPT), d'une segmentation hiérarchique issue des précédentes coupes et enfin d'un exemple de niveau de détail observé suivant une hiérarchie donnée.

alors de calculer une distance / similarité entre deux régions voisines N_i et N_j. La procédure de construction d'un BPT $\mathfrak{T}$ est illustrée par l'algorithme 1. Il nécessite la définition d'un graphe d'adjacence de régions $\mathfrak{G}$ qui permet à chaque région (ou nœud) de connaître ses régions voisines au cours de chaque étape menant à la création du BPT $\mathfrak{T}$. Les nœuds du graphe $\mathfrak{G}$ correspondent aux nœuds du BTP $\mathfrak{T}$ à une étape t de sa construction et les arêtes correspondent aux adjacences A_ij liant les régions voisines N_i et N_j. Chaque lien d'adjacence A_ij est associé à une valeur correspondant à la distance de similarité entre N_i et N_j. Afin d'identifier la sélection des deux régions à fusionner, une liste ordonnée $\mathcal{W}$ contient les adjacences de l'étape courante. Ainsi, si A_ij est associée à la valeur de similarité la plus optimale, les régions N_i et N_j sont sélectionnées. Tant que la racine du BPT $\mathfrak{T}$ n'est pas obtenue (i.e. il ne reste plus qu'un seul nœud dans dans le graphe $\mathfrak{G}$), les régions voisines les plus proches N_i et N_j sont fusionnées menant à la création d'une nouvelle région N_{ij}. Cette dernière doit alors remplacer ses régions filles N_i et N_j dans le graphe d'adjacence des régions $\mathfrak{G}$ qui doit être mis à jour. En effet, le modèle $M_r(N_{ij})$ doit être défini et les distances entre cette nouvelle région et les voisines de N_i et de N_j sont à recalculer. La liste de priorité $\mathcal{W}$ doit également être mise à jour. Une fois que la racine est identifiée, l'algorithme arrive à terme et le BPT $\mathfrak{T}$ est créé.

Data: Image I, Modèle $M_r(N_i)$, Critère de fusion $O_r(N_i, N_j)$
Result: BPT $\mathfrak{T}$
Déterminer la partition initiale Ω à partir de I;
Créer un graph d'adjacence de régions $\mathfrak{G}$; `/* calculer les distances entre les régions voisines */`
Initialiser et ordonner la liste $\mathcal{W}$ de priorité des fusions;
while *racine pas obtenue* **do**
  Déterminer à partir de $\mathcal{W}$ les deux régions les plus proches N_i et N_j;
  Fusionner N_i et N_j et le ajouter nouveau nœud N_ij à $\mathfrak{T}$; `/* calculer le modèle` $M_r(N_ij)$ `*/`
  Mettre à jour $\mathfrak{G}$; `/* calculer la distance entre` N_ij `et les voisins de` N_i `et de` N_j `*/`
  Mettre à jour et ordonner la liste de priorité $\mathcal{W}$;
  if *Il reste une seule région* **then**
    racine ← région restante ; `/* support de l'image */`
  end
end
$\mathfrak{T} \leftarrow$ *racine*;

Algorithm 1: Algorithme de construction d'un BPT

Afin d'assurer la pertinence et la cohérence des différentes régions du BPT $\mathfrak{T}$, l'ordre de fusion observé au cours de sa création est important. De ce fait, cet algorithme reste fortement séquentiel car le choix du couple de régions à fusioner à un instant t dépend considérablement du choix effectué au temps $t-1$. Bien que le BPT reste une structure hiérarchique intéressante dans le domaine de l'analyse et du traitement d'images, sa création peut être coûteuse tant en temps d'exécution qu'en mémoire, puisque sa complexité est quadratique. Des problématiques de passage à l'échelle peuvent alors défavoriser le choix sur l'utilisation du BPT face à d'autres structures qui sont moins coûteuses. En effet, la création d'un BPT à partir d'une image de grande dimension (e.g. 11.000 x 11.000 pixels) peut être très douloureuse. De telles images sont fréquentes surtout dans le domaine de la télédétection.

Objectif

Face à la problématique liée à la difficulté de passage à l'échelle de la construction d'un BPT, l'objectif principal de ce stage est de proposer un algorithme parallèle / distribué afin de minimiser les différents coûts liés à l'exécution et à la mémoire. L'ordre de fusion des différentes régions étant très important et n'étant pas connu à l'avance, la construction d'un BPT se base sur un algorithme fortement séquentiel.

Plusieurs travaux comme [10] et [11] traitent le cas du clustering hiérarchique où le dendrogramme résultant est très similaire au BPT. Ces travaux se basent sur le principe d'arbre couvrant minimal MST (*Minimum Spanning Tree*). Il a, en effet, été démontré dans [12] que le calcul d'un clustering hiérarchique se basant sur une métrique *single-linkage* (pour la distance de similarité) peut être réduit au calcul du MST correspondant. Il y a beaucoup de travaux [13, 14, 15] qui traitent et qui proposent des algorithmes parallèles / distribués pour le calcul d'un MST. Dans le cas du BPT, le choix d'une métrique *single-linkage* est possible mais pas forcément suffisant afin d'avoir une structure hiérarchique cohérente vis-à-vis de la scène observée. En effet, la force d'un BPT réside dans le fait que l'expert puisse injecter, à travers le choix d'une métrique de similarité entre régions, une connaissance issue d'une expertise du domaine étudié. D'autres travaux [16, 17, 18] proposent, quant-à eux, des alternatives de parallélisation qui mènent à l'obtention d'un clustering hiérarchique différent de celui que l'on a à partir d'un algorithme séquentiel. Dans ce cas, il est donc nécéssaire de bien démontrer que le résutat obtenu est proche ou similaire au résultat attendu et que la pertinence de la structure hiérarchique n'en est pas moindre.

Pour répondre à la problématique du sujet, différentes pistes sont donc à suivre en se basant sur les différents travaux de l'état de l'art. Quelques questions sont donc à soulever :

- Y a-t-il encore un verrou pour la parallélisation / distribution de la construction d'un BPT ?
- Si oui, l'objectif du stage sera de proposer un algorithme parallèle / distribué
- Si non, l'objectif du stage sera de déterminer quelle approche est la plus pertinente parmis celles de l'état de l'art, et de l'implémenter.

Profil

Le candidat ou la candidate doit être d'un niveau M2 ou équivalent (dernière année d'école d'ingénieur) en informatique et posséder de solides compétences en programmation parallèle ainsi que des bases en traitement d'images. Les développements s'effectueront en Python pour la partie prototypage et C++ pour la partie implémentation efficace.

Le stage sera basé au LRE (Laboratoire de Recherche de l'EPITA, `https://www.lre.epita.fr/image/`), à Strasbourg. Des visites à Paris au cours du stage pourront être envisagées.

Contact et candidature

Les encadrants du stage sont :

- Guillaume Tochon, Enseignant-Chercheur, LRE, EPITA Paris, France.
- Edwin Carlinet, Enseignant-Chercheur, LRE, EPITA Paris, France.
- Jimmy Randrianasoa, Enseignant-Chercheur, LRE, EPITA Strasbourg, France.

Est également impliqué sur le projet :

- Benoît Naegel, Professeur, Université de Strasbourg, ICube, Illkirch-Graffenstaden, France.

L'équipe encadrante possède une grande expérience du traitement d'images et des méthodes de programmation parallèle appliquées à ces problématiques.

Rémunération 1000€brut/mois
Début du stage À partir de mi-février
Durée du stage 5/6 mois

Merci d'envoyer votre candidature (CV, lettre de motivation, relevé de notes du M1/2^e année de cycle ingénieur + descriptif du M2/3^e année de cycle ingénieur) à {guillaume.tochon,edwin.carlinet,jimmy.randrianasoa}@lre.epita.fr.

References

[1] P. Salembier and L. Garrido, "Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval," *IEEE Transactions on Image Processing*, vol. 9, no. 4, pp. 561–576, 2000.

[2] V. Vilaplana, F. Marques, and P. Salembier, "Binary partition trees for object detection," *IEEE Transactions on Image Processing*, vol. 17, no. 11, pp. 2201–2216, 2008.

[3] S. Valero, P. Salembier, and J. Chanussot, "Hyperspectral image representation and processing with binary partition trees," *IEEE Transactions on Image Processing*, vol. 22, no. 4, pp. 1430–1443, 2013.

[4] M. A. Veganzones, G. Tochon, M. Dalla-Mura, A. J. Plaza, and J. Chanussot, "Hyperspectral image segmentation using a new spectral unmixing-based binary partition tree representation," *IEEE Transactions on Image Processing*, vol. 23, no. 8, pp. 3574–3589, 2014.

[5] P. Salembier and S. Foucher, "Optimum graph cuts for pruning binary partition trees of polarimetric sar images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 9, pp. 5493–5502, 2016.

[6] P. Salembier, A. Oliveras, and L. Garrido, "Antiextensive connected operators for image and sequence processing," *IEEE Transactions on Image Processing*, vol. 7, no. 4, pp. 555–570, 1998.

[7] E. Carlinet and T. Geraud, "Getting a morphological tree of shapes for multivariate images: Paths, traps, and pitfalls," in *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 615–619, 2014.

[8] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[9] V. Machairas, M. Faessel, D. Cárdenas-Peña, T. Chabardes, T. Walter, and E. Decencière, "Waterpixels," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 3707–3716, 2015.

[10] C. F. Olson, "Parallel algorithms for hierarchical clustering," *Parallel Computing*, vol. 21, no. 8, pp. 1313–1325, 1995.

[11] S. Rajasekaran, "Efficient parallel hierarchical clustering algorithms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 6, pp. 497–502, 2005.

[12] J. C. Gower and G. J. S. Ross, "Minimum spanning trees and single linkage cluster analysis," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 18, no. 1, pp. 54–64, 1969.

[13] D. A. Bader and G. Cong, "Fast shared-memory algorithms for computing the minimum spanning forest of sparse graphs," *Journal of Parallel and Distributed Computing*, vol. 66, no. 11, pp. 1366–1378, 2006.

[14] H. Karloff, S. Suri, and S. Vassilvitskii, "A model of computation for mapreduce," in *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pp. 938–948, 2010.

[15] A. Andoni, A. Nikolov, K. Onak, and G. Yaroslavtsev, "Parallel algorithms for geometric graph problems," in *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pp. 574–583, 2014.

[16] M. Dash, S. Petrutiu, and P. Scheuermann, "Efficient parallel hierarchical clustering," in *Euro-Par 2004 Parallel Processing*, pp. 363–371, 2004.

[17] S. Lattanzi, T. Lavastida, K. Lu, and B. Moseley, "A framework for parallelizing hierarchical clustering methods," in *Machine Learning and Knowledge Discovery in Databases*, pp. 73–89, 2020.

[18] J. Lefèvre, J. Cousty, B. Perret, and H. Phelippeau, "Join, select, and insert: Efficient out-of-core algorithms for hierarchical segmentation trees," in *Discrete Geometry and Mathematical Morphology*, pp. 274–286, 2022.