Un nouvel algorithme efficace de *Split & Merge* pour systèmes embarqués

COMPAS 2021

N. Maurice, J. Sopena, L. Lacassagne

7 juillet 2021

Introduction

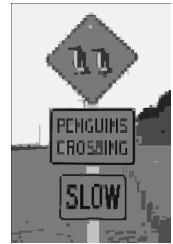
Traitement image : de Hong Kong à la conduite autonome

Introduction

Traitement image : de Hong Kong à la conduite autonome

Progrès importants récents grâce à l'IA :

1. pré-traitement pour extraire zones homogènes
2. segmentation par réseaux de neurones



Introduction

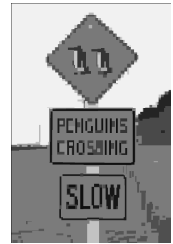
Traitement image : de Hong Kong à la conduite autonome

Progrès importants récents grâce à l'IA :

1. pré-traitement pour extraire zones homogènes
2. segmentation par réseaux de neurones

Les solutions actuelles :

- trop lent/trop lourd : pas adapté à l'embarqué
- trop irrégulier : pas adapté au temps réel



Introduction

Traitement image : de Hong Kong à la conduite autonome

Progrès importants récents grâce à l'IA :

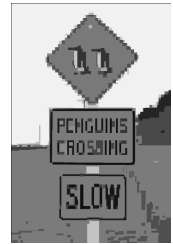
1. pré-traitement pour extraire zones homogènes
2. segmentation par réseaux de neurones

Les solutions actuelles :

- trop lent/trop lourd : pas adapté à l'embarqué
- trop irrégulier : pas adapté au temps réel

Objectifs : un nouvel algorithme de *Split & Merge* tel que

- cadence temps réel : 25 images/s
- temps d'execution indépendant des critères de fusion et du contenu de l'image
- pas de compromis sur la qualité



Fonctionnement du *Split & Merge*

Objectif : Obtenir des **zones homogènes** continues, càd dont la variance est inférieure à une borne donnée

Split & Merge composé de deux étapes [Aneja] :

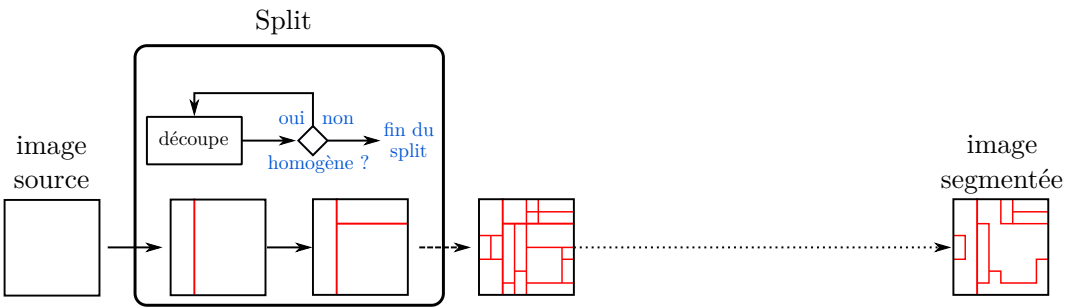
- ***Split*** : découpe récursive de l'image en rectangle (sur-segmentation)
- ***Merge*** : fusion des régions voisines semblables

Fonctionnement du *Split & Merge*

Objectif : Obtenir des **zones homogènes** continues, càd dont la variance est inférieure à une borne donnée

Split & Merge composé de deux étapes [Aneja] :

- ***Split*** : découpe récursive de l'image en rectangle (sur-segmentation)
- ***Merge*** : fusion des régions voisines semblables

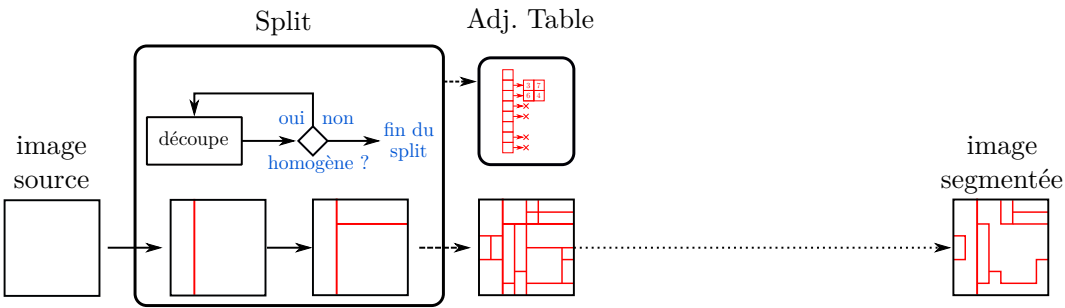


Fonctionnement du *Split & Merge*

Objectif : Obtenir des **zones homogènes** continues, càd dont la variance est inférieure à une borne donnée

Split & Merge composé de deux étapes [Aneja] :

- ***Split*** : découpe récursive de l'image en rectangle (sur-segmentation)
- ***Merge*** : fusion des régions voisines semblables

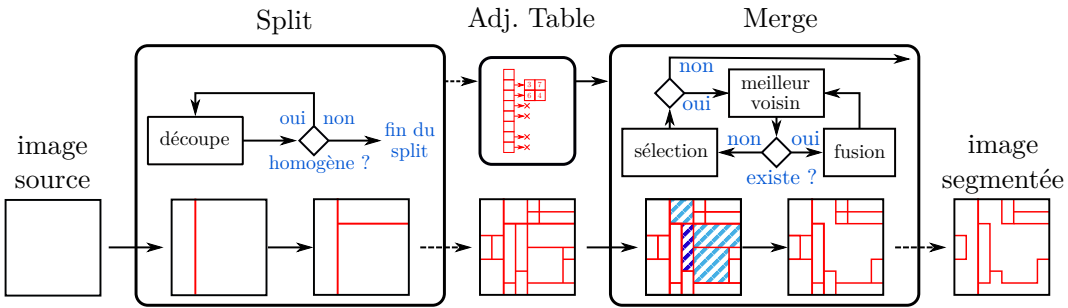


Fonctionnement du *Split & Merge*

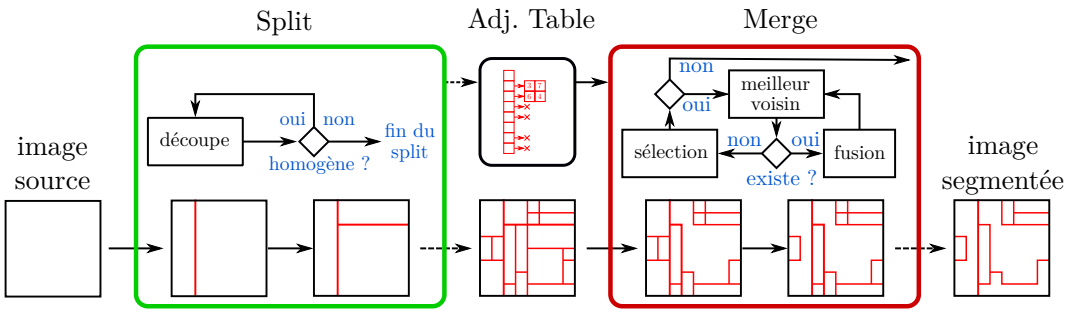
Objectif : Obtenir des **zones homogènes** continues, càd dont la variance est inférieure à une borne donnée

Split & Merge composé de deux étapes [Aneja] :

- ***Split*** : découpe récursive de l'image en rectangle (sur-segmentation)
- ***Merge*** : fusion des régions voisines semblables



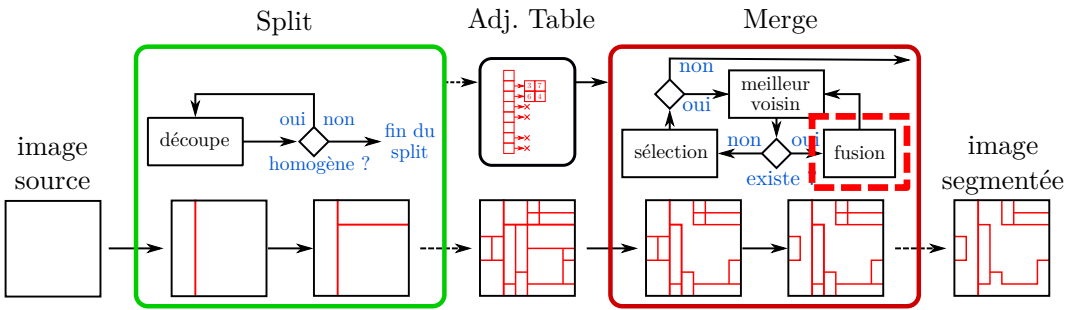
Problème des rélallocations du Merge



temps exécution dépend peu des données et paramètres

temps exécution dépend beaucoup des données et paramètres

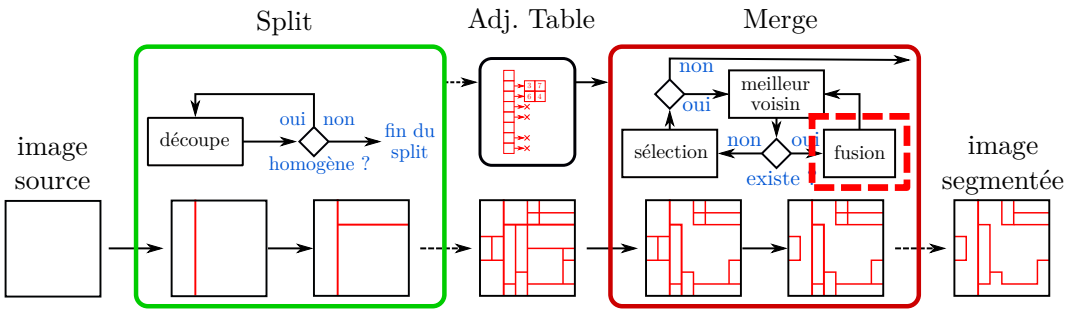
Problème des rélallocations du Merge



temps exécution dépend peu des données et paramètres

temps exécution dépend beaucoup des données et paramètres

Problème des rélallocations du Merge



Problème 1 : Réallocations lors des fusions : Utilisation d'un TTA

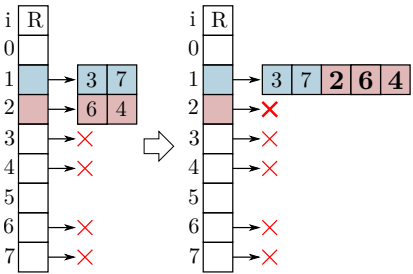
Problème des rélallocations du Merge

Fusions de régions dans la table d'adjacence :

⇒ Concaténations de tableaux

⇒ Réallocations mémoires

⇒ Fragmentation mémoire



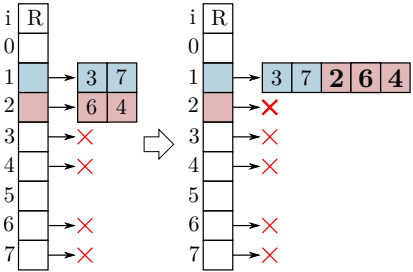
Problème des rélallocations du Merge

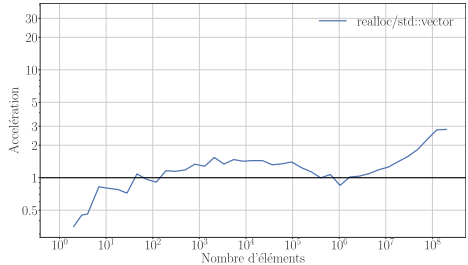
Fusions de régions dans la table d'adjacence :

⇒ Concaténations de tableaux

⇒ Réallocations mémoires

⇒ Fragmentation mémoire





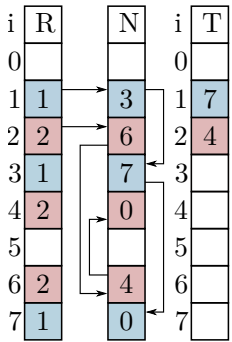
Méthode par sur-allocation mise en oeuvre dans `std::vector`

⇒ pas de gains dans l'intervalle visé ($\approx 10^6$)

Suppression des réallocations dans les fusions

Solution : Structure de TTA

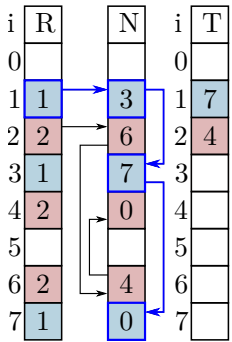
3 tableaux : R (Root), N (Next), T (Tail). 1 entrée = 1 région initiale



Suppression des réallocations dans les fusions

Solution : Structure de TTA

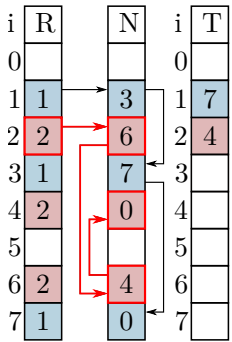
3 tableaux : R (Root), N (Next), T (Tail). 1 entrée = 1 région initiale



Suppression des réallocations dans les fusions

Solution : Structure de TTA

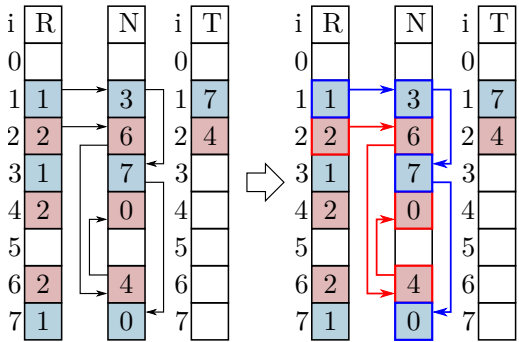
3 tableaux : R (Root), N (Next), T (Tail). 1 entrée = 1 région initiale



Suppression des réallocations dans les fusions

Solution : Structure de TTA

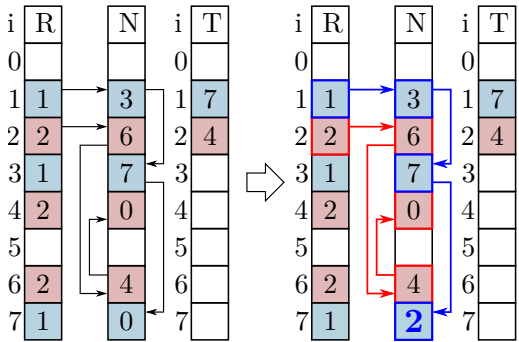
3 tableaux : R (Root), N (Next), T (Tail). 1 entrée = 1 région initiale



Suppression des réallocations dans les fusions

Solution : Structure de TTA

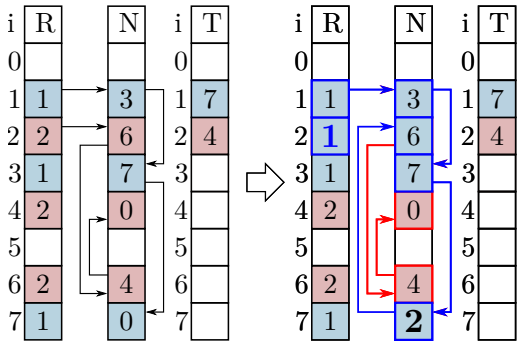
3 tableaux : R (Root), N (Next), T (Tail). 1 entrée = 1 région initiale



Suppression des réallocations dans les fusions

Solution : Structure de TTA

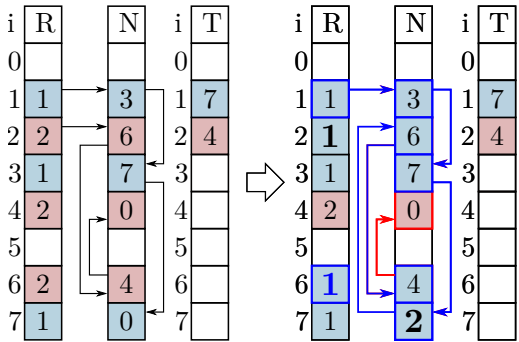
3 tableaux : R (Root), N (Next), T (Tail). 1 entrée = 1 région initiale



Suppression des réallocations dans les fusions

Solution : Structure de TTA

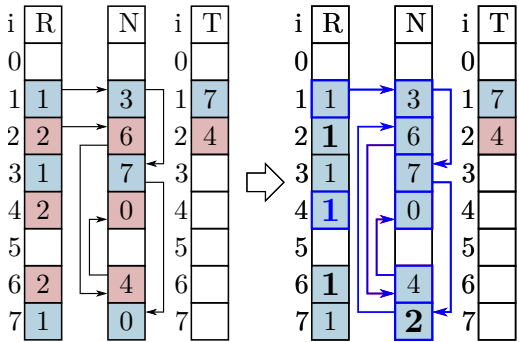
3 tableaux : R (Root), N (Next), T (Tail). 1 entrée = 1 région initiale



Suppression des réallocations dans les fusions

Solution : Structure de TTA

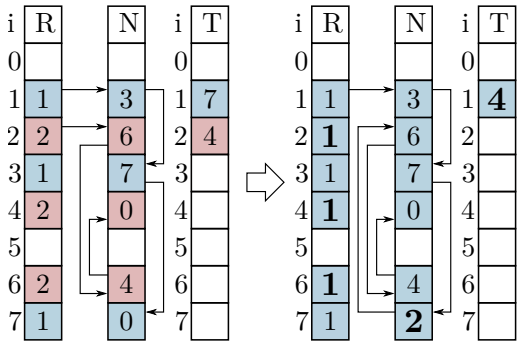
3 tableaux : R (Root), N (Next), T (Tail). 1 entrée = 1 région initiale



Suppression des réallocations dans les fusions

Solution : Structure de TTA

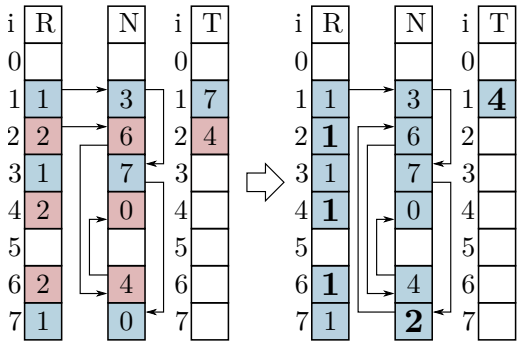
3 tableaux : R (Root), N (Next), T (Tail). 1 entrée = 1 région initiale

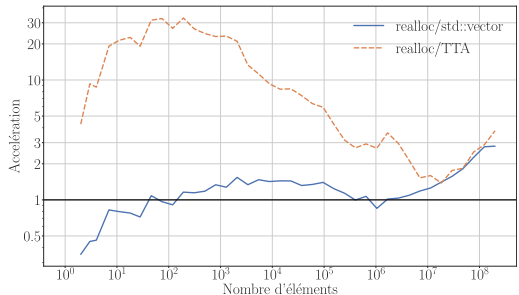


Suppression des réallocations dans les fusions

Solution : Structure de TTA

3 tableaux : R (Root), N (Next), T (Tail). 1 entrée = 1 région initiale

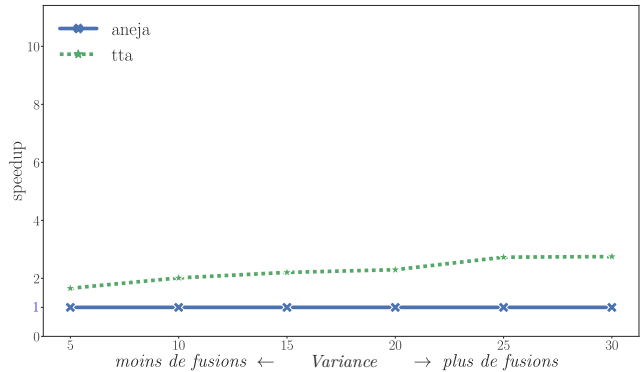




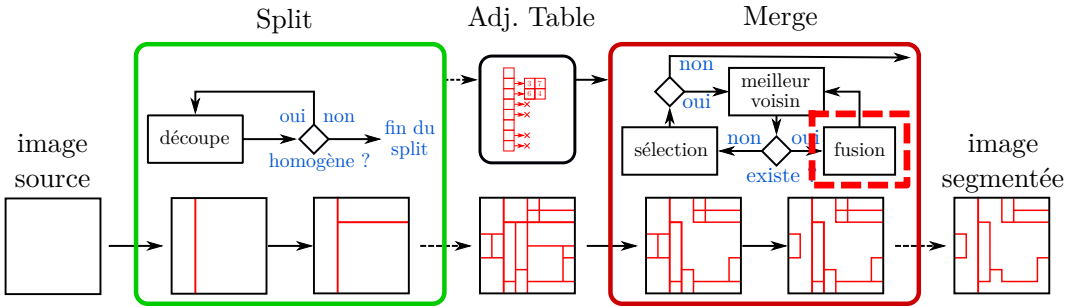
Plus besoins de réallocations $\Rightarrow$ accelération importante dans intervalle visé ($\approx 10^6$)

Mesures des segmentations sur système embarqué

Test sur 8 Images routières (*CAMVID*) de 960 $\times$ 720
Mesures sur *Nvidia Jetson NX* (*Cortex A76* custom) de 15 Watts



Supression des réallocations

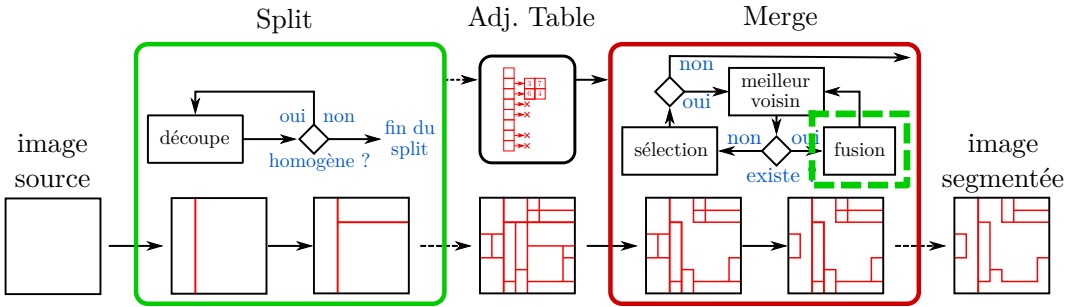


temps exécution dépend peu des données et paramètres

temps exécution dépend beaucoup des données et paramètres

Problème 1 : Réallocations lors des fusions : Utilisation d'un TTA

Supression des réallocations

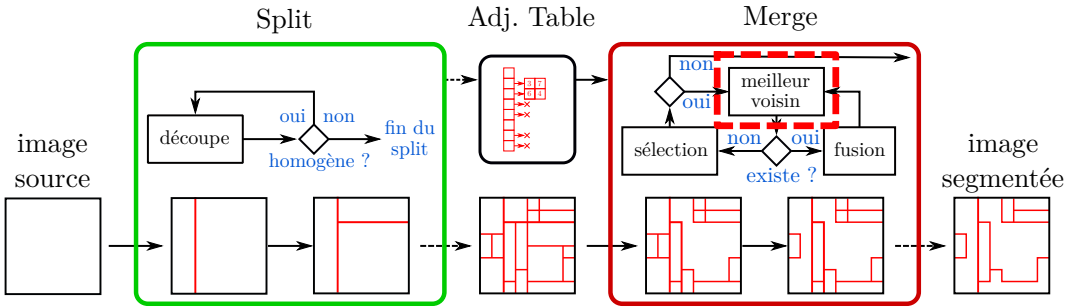


temps exécution dépend peu des données et paramètres

temps exécution dépend beaucoup des données et paramètres

Problème 1 : Réallocations lors des fusions : Utilisation d'un TTA

Supression des réallocations



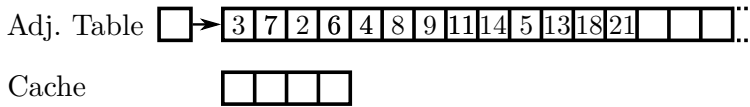
temps exécution dépend peu des données et paramètres

temps exécution dépend beaucoup des données et paramètres

Problème 1 : Réallocations lors des fusions : Utilisation d'un TTA
Problème 2 : Perte de localité mémoire lors du parcours du TTA

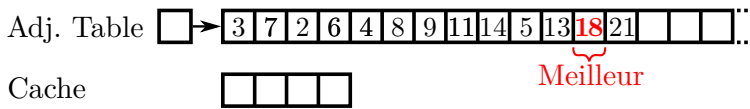
Regain de localité dans la recherche de meilleur voisin

Cache logiciel : Un « bon »voisin va souvent le rester à l'étape suivante



Regain de localité dans la recherche de meilleur voisin

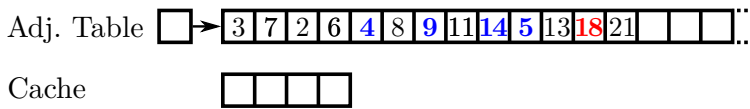
Cache logiciel : Un « bon »voisin va souvent le rester à l'étape suivante



1. Recherche meilleur voisin dans voisinage

Regain de localité dans la recherche de meilleur voisin

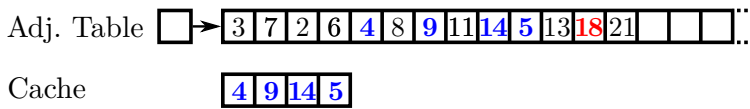
Cache logiciel : Un « bon »voisin va souvent le rester à l'étape suivante



1. Recherche meilleur voisin dans voisinage
2. Insertions des « bons »voisins dans le cache

Regain de localité dans la recherche de meilleur voisin

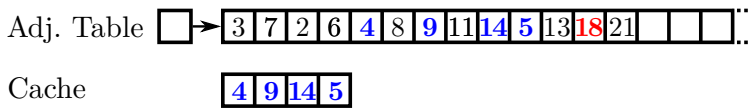
Cache logiciel : Un « bon »voisin va souvent le rester à l'étape suivante



1. Recherche meilleur voisin dans voisinage
2. Insertions des « bons »voisins dans le cache

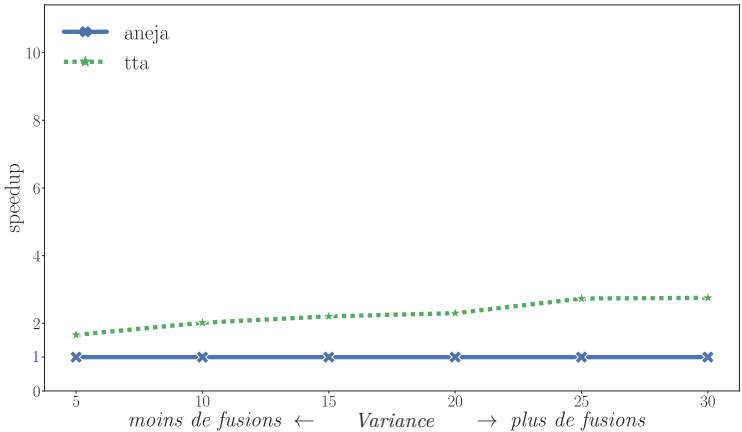
Regain de localité dans la recherche de meilleur voisin

Cache logiciel : Un « bon »voisin va souvent le rester à l'étape suivante

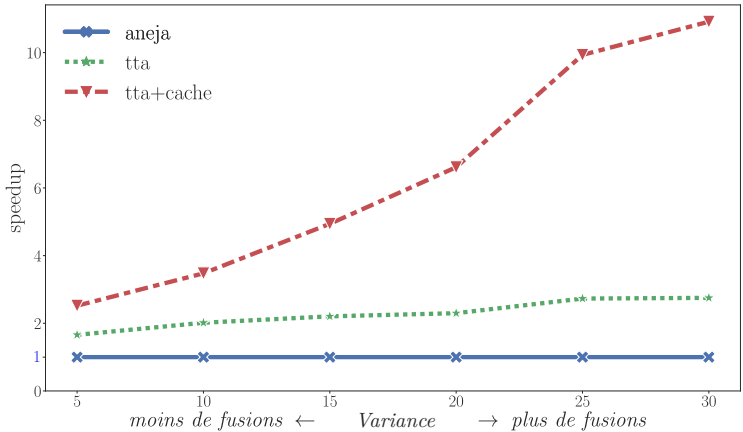


1. Recherche meilleur voisin dans voisinage
2. Insertions des « bons »voisins dans le cache
3. Fusions itératives dans le cache et insertion des nouveaux voisins dans le voisinage

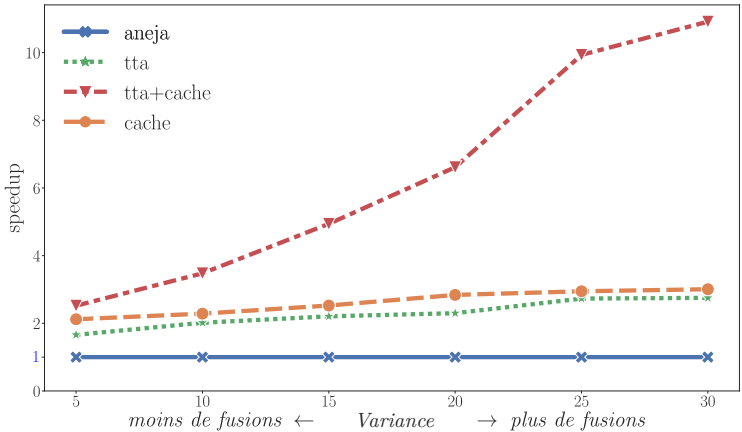
Evaluation performances du cache logiciel



Evaluation performances du cache logiciel

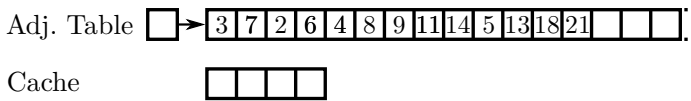


Evaluation performances du cache logiciel



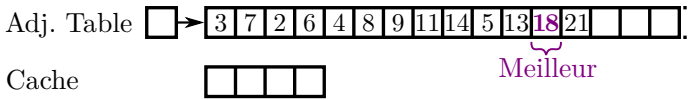
Un cache à une passe & supression des cas pathologiques

La recherche avec cache nécessite 2 parcours : Est-il possible de faire avec un seul ?



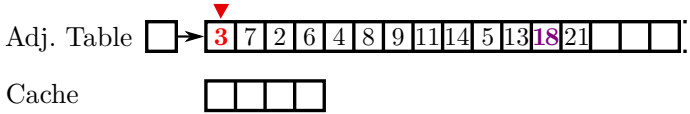
Un cache à une passe & supression des cas pathologiques

La recherche avec cache nécessite 2 parcours : Est-il possible de faire avec un seul ?



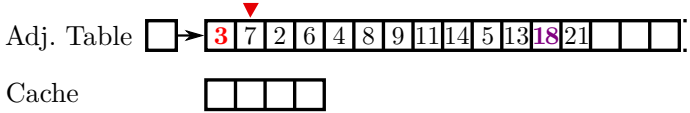
Un cache à une passe & supression des cas pathologiques

La recherche avec cache nécessite 2 parcours : Est-il possible de faire avec un seul ?



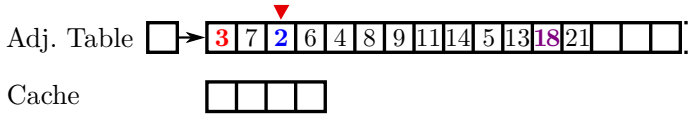
Un cache à une passe & supression des cas pathologiques

La recherche avec cache nécessite 2 parcours : Est-il possible de faire avec un seul ?



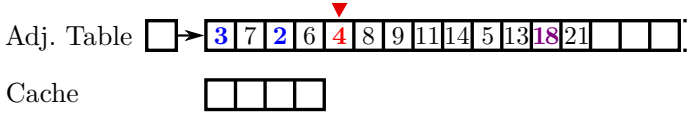
Un cache à une passe & supression des cas pathologiques

La recherche avec cache nécessite 2 parcours : Est-il possible de faire avec un seul ?



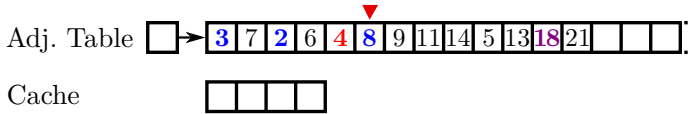
Un cache à une passe & supression des cas pathologiques

La recherche avec cache nécessite 2 parcours : Est-il possible de faire avec un seul ?



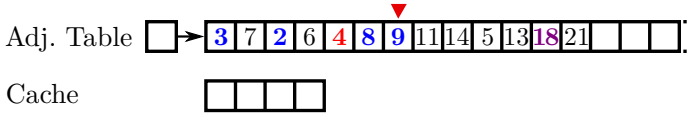
Un cache à une passe & supression des cas pathologiques

La recherche avec cache nécessite 2 parcours : Est-il possible de faire avec un seul ?



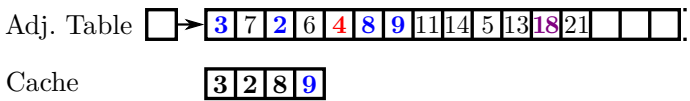
Un cache à une passe & supression des cas pathologiques

La recherche avec cache nécessite 2 parcours : Est-il possible de faire avec un seul ?



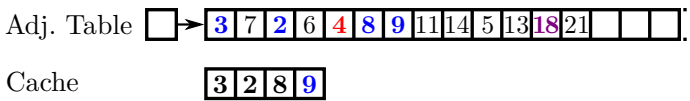
Un cache à une passe & supression des cas pathologiques

La recherche avec cache nécessite 2 parcours : Est-il possible de faire avec un seul ?



Un cache à une passe & supression des cas pathologiques

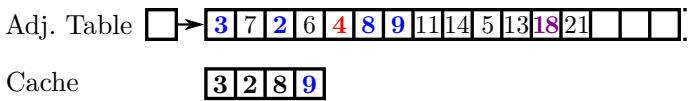
La recherche avec cache nécessite 2 parcours : Est-il possible de faire avec un seul ?



Si les nouveaux « bons »voisins sont insérés à la fin ⇒ cas pathologique

Un cache à une passe & supression des cas pathologiques

La recherche avec cache nécessite 2 parcours : Est-il possible de faire avec un seul ?

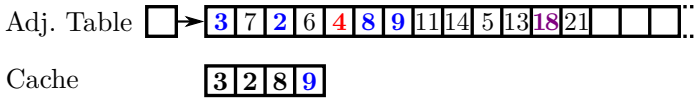


Si les nouveaux « bons »voisins sont insérés à la fin ⇒ cas pathologique

Solution : Inverser sens de parcours à chaque étape

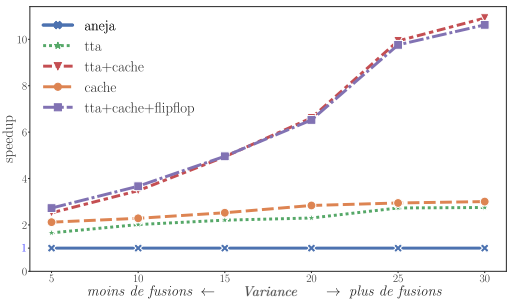
Un cache à une passe & supression des cas pathologiques

La recherche avec cache nécessite 2 parcours : Est-il possible de faire avec un seul ?



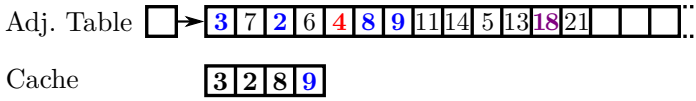
Si les nouveaux « bons »voisins sont insérés à la fin ⇒ cas pathologique

Solution : Inverser sens de parcours à chaque étape



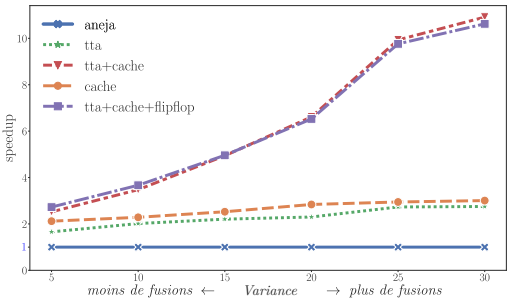
Un cache à une passe & supression des cas pathologiques

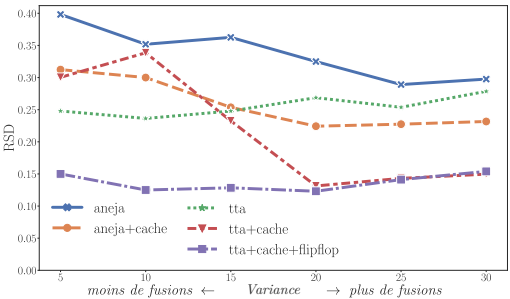
La recherche avec cache nécessite 2 parcours : Est-il possible de faire avec un seul ?



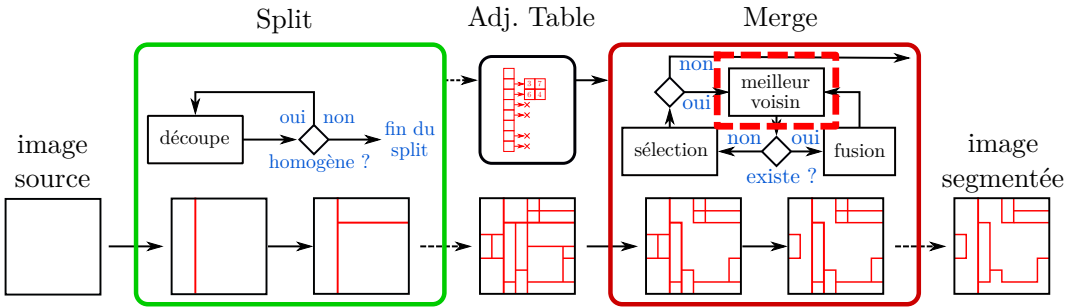
Si les nouveaux « bons »voisins sont insérés à la fin ⇒ cas pathologique

Solution : Inverser sens de parcours à chaque étape





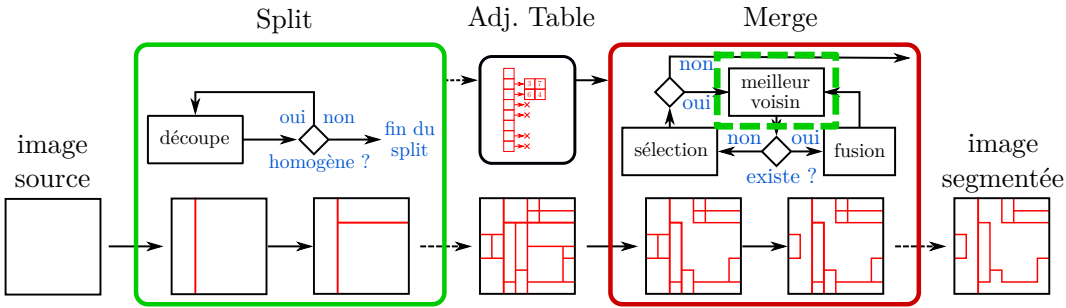
Regain de localité mémoire & flipflop



Problème 1 : Réallocations lors des fusions : Utilisation d'un TTA

Problème 2 : Perte de localité mémoire lors du parcours du TTA : Stratégie de cache

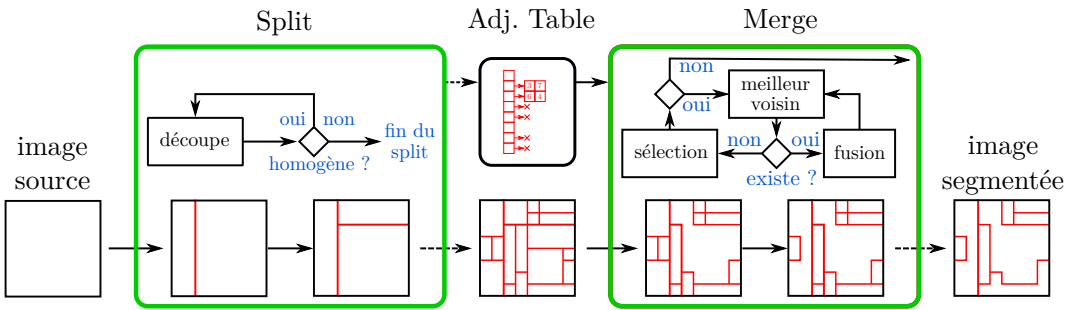
Regain de localité mémoire & flipflop



Problème 1 : Réallocations lors des fusions : Utilisation d'un TTA

Problème 2 : Perte de localité mémoire lors du parcours du TTA : Stratégie de cache

Regain de localité mémoire & flipflop

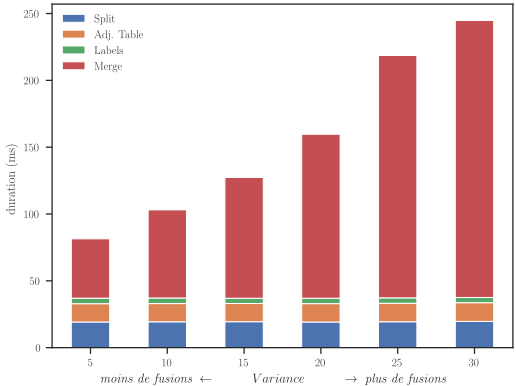


Problème 1 : Réallocations lors des fusions : Utilisation d'un TTA

Problème 2 : Perte de localité mémoire lors du parcours du TTA : Stratégie de cache

Regularité en fonction de la variance

Aneja

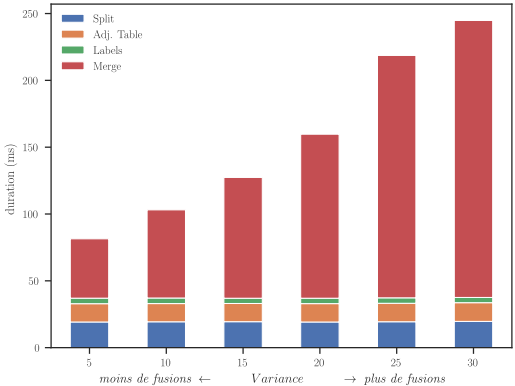


4 à 12 images/s

dépend fortement de la variance choisie

Regularité en fonction de la variance

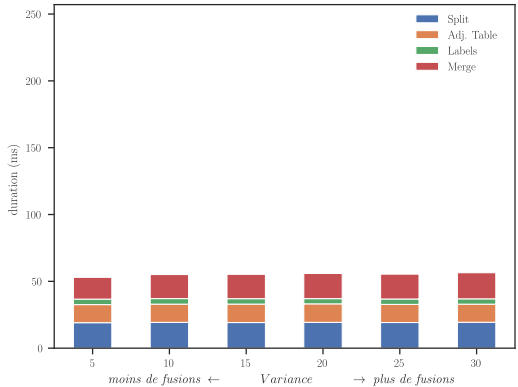
Aneja



4 à 12 images/s

dépend fortement de la variance choisie

TTA+Cache+Flipflop

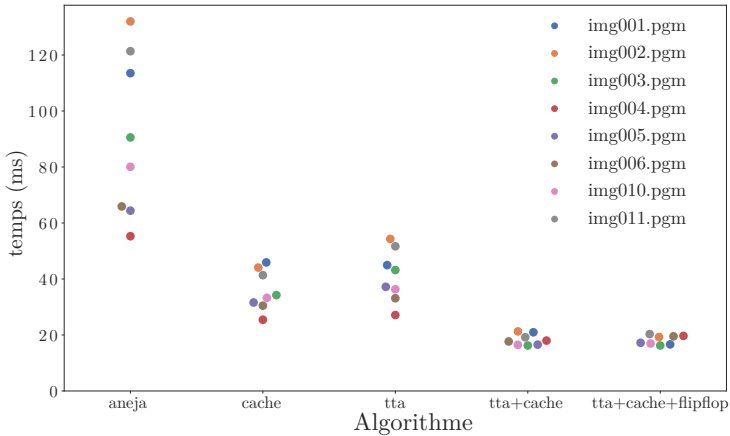


≈ 20 images/s

dépend peu de la variance choisie

Regularité sur plusieurs images

Est ce régulier sur plusieurs images à Variance fixe ?



Conclusions

Nouvel algoritme résolvant des problèmes systèmes liés a l'utilisation de la mémoire

- rapide : 20 images/s atteint
- régulier (varie peu en fonction de la variance et du contenu)
- pas de compromis sur la qualité de la segmentation (PSNR et SSIM)

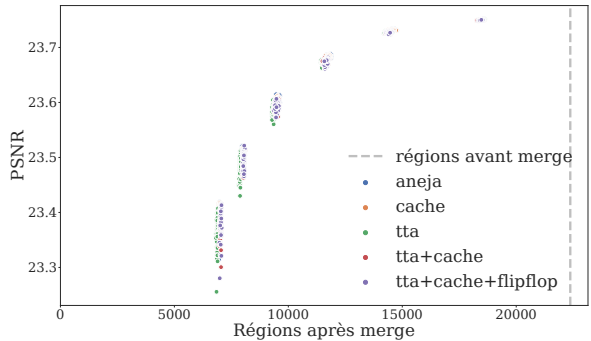
Future works :

- pipeline du Split avec le Merge pour atteindre 25 images/s
- intégration chaine de traitement complète avec IA
- comparaison avec IA

Vérification de la qualité de segmentation

Cache & Flip-flop ne produisent pas la même segmentation que Aneja
⇒ besoin de vérifier la segmentation
Mesure de la distance entre segmentation obtenue et image d'origine : PSNR et SSIM

Groupes de points proches de Aneja
⇒ segmentation similaire



Base d'images CAMVID

img001

img006