UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

ÉVALUATION DE TECHNIQUES D'INDEXAGE POUR LA FUSION DE REPRÉSENTATIONS

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

PAR
NICOLAS THERRIEN

MAI 2007

UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

Département d'informatique et d'ingénierie

Ce mémoire intitulé :

ÉVALUATION DE TECHNIQUES D'INDEXAGE POUR LA FUSION DE REPRÉSENTATIONS

présenté par
Nicolas Therrien

pour l'obtention du grade de maître ès science (M.Sc.)

a été évalué par un jury composé des personnes suivantes :

Marek Zaremba ..... Directeur de recherche
Karim El Guemhioui ..... Président du jury
Jurek Czyzowicz ..... Membre du jury

Mémoire accepté le : 24 mai 2007

J'adresse mes remerciements à Marek Zaremba pour son soutien tout au long de mes études et à tous les membres de l'équipe du Laboratoire des Systèmes Spatiaux Intelligents avec qui il a été très agréable de travailler.

Table des matières

Liste des figures

Liste des tableaux

Résumé

La fusion de données (data fusion) est un processus de raffinement des données brutes pour obtenir une information de meilleure qualité. Elle fait l'objet d'études très diversifiées telles que l'amélioration de la vision nocturne, la synthèse d'analyses provenant de capteurs multiples et l'amélioration de la qualité d'images basses résolutions (pansharpening).

Ce mémoire traite plus particulièrement de la fusion de représentations (fusion of representations). En télédétection, il peut être intéressant d'extraire de l'information de plusieurs sources différentes. Ce faisant, il arrive que les sources utilisées ne soient pas directement compatibles à cause de leurs résolutions différentes, de leurs tailles différentes ou encore parce que leur type de données est fondamentalement différent (matrice, nuage de points, vecteurs...). L'élaboration d'une solution qui permettrait l'utilisation de ces données de manière transparente est désirable. Pour faire face à ce problème, nous abordons deux aspects qui en découlent : la fusion de représentations et la plateforme dans laquelle elle est implantée. Nous nous intéressons particulièrement aux performances du choix d'un index spatial (spatial index) sur la fusion de représentations, dans un contexte de prototypage rapide en télédétection.

Abstract

Data fusion is a data refinement process that aims at obtaining information of greater quality out of raw data. It is used in various studies such as night vision enhancement, multiple sensor analysis synthesis and the enhancement of low resolution images (pansharpening).

In this research project, we focus more specifically on the fusion of representations. In remote sensing, it may be interesting to extract information from multiple sources. When doing so, chances are that the sources being used are not directly compatible because of different resolutions, different sizes or even because their data types are intrinsically different (matrix, points, vectors...). The development of a solution which would allow the use of these data in a transparent way is desirable. In our approach to this problem, we consider two aspects that result from it: the fusion of representations and the framework in which it is used. More specifically, we are sudying the impact of the spatial index on the performance of the fusion of representations in a context of rapid prototyping in remote sensing.

1 Introduction

Ce mémoire se situe dans le cadre de la fusion de données (*Data Fusion*) dans le domaine de la télédétection et traite plus particulièrement de la fusion de représentations (*Fusion of Representations*). La fusion de représentations est une branche de la fusion de données qui concerne les méthodes d'intégration des données spatiales disparates en un modèle uniforme qui puisse faciliter l'accès aux données.

Au cours de nos recherches, un moteur de fusion de représentations a été développé pour nous permettre d'effectuer la fusion de représentations de données provenant d'images satellitaires et d'images lidar. Il a été identifié que le module ayant le plus d'impact sur les performances de ce moteur était l'index spatial. Ce mémoire est dédié à l'étude de différentes techniques d'indexage spatial afin de déterminer quelle serait la méthode la plus appropriée dans un contexte de prototypage des algorithmes en télédétection..

En télédétection, la fusion de représentations est le plus souvent trouvée à même les logiciels géomatiques utilisés pour la visualisation et le traitement des données. Chaque logiciel présente et traite les données dans un cadre bien précis défini par le concepteur. Ce cadre d'opération est du ressort de la fusion de représentations et permet de faire le lien entre ce que l'utilisateur voit et ce que les données sont en réalité. Comme le moteur d'affichage et de traitement des données géospatiales est le fondement de tout logiciel de géomatique, la fusion de représentations en télédétection est devenue un champ d'études propriétaire et relativement secret. La fusion de représentations n'en reste pas moins inflexible aux yeux des utilisateurs en ce sens qu'ils n'ont d'autre choix que celui du logiciel à acheter : il faut choisir le logiciel qui utilise une fusion des représentations qui soit la mieux appropriée au travail à effectuer. C'est pour cette raison, nous le verrons plus loin, que certains logiciels sont reconnus comme étant plus performants avec les

formats matriciels et d'autres avec les formats vectoriels et ponctuels. Une comparaison entre les logiciels de géomatique sera présentée pour faire ressortir non seulement leurs avantages et leurs inconvénients, mais surtout la raison pour laquelle il est intéressant de construire une plateforme spécialisée dans le prototypage des algorithmes de télédétection.

À cause de cette inflexibilité des logiciels commerciaux, nous avons créé une plateforme de base pour la manipulation des données géomatiques dans laquelle nous avons intégré le moteur de fusion de représentations. C'est à l'aide de cette plateforme, que nous avons nommée Feature Ex-RA, qu'ont été menés les tests qui nous ont permis d'évaluer les différentes techniques d'indexage.

Un exemple pratique de la fusion de représentations est la technique du pansharpening. On l'utilise pour améliorer la qualité d'images basse résolution en fusionnant une image de mauvaise qualité avec une image haute résolution. On la retrouve plus particulièrement dans le domaine de l'imagerie satellitaire où l'on effectue la fusion de la bande panchromatique avec les bandes spectrales pour obtenir une image de meilleure qualité (voir Figure 1).

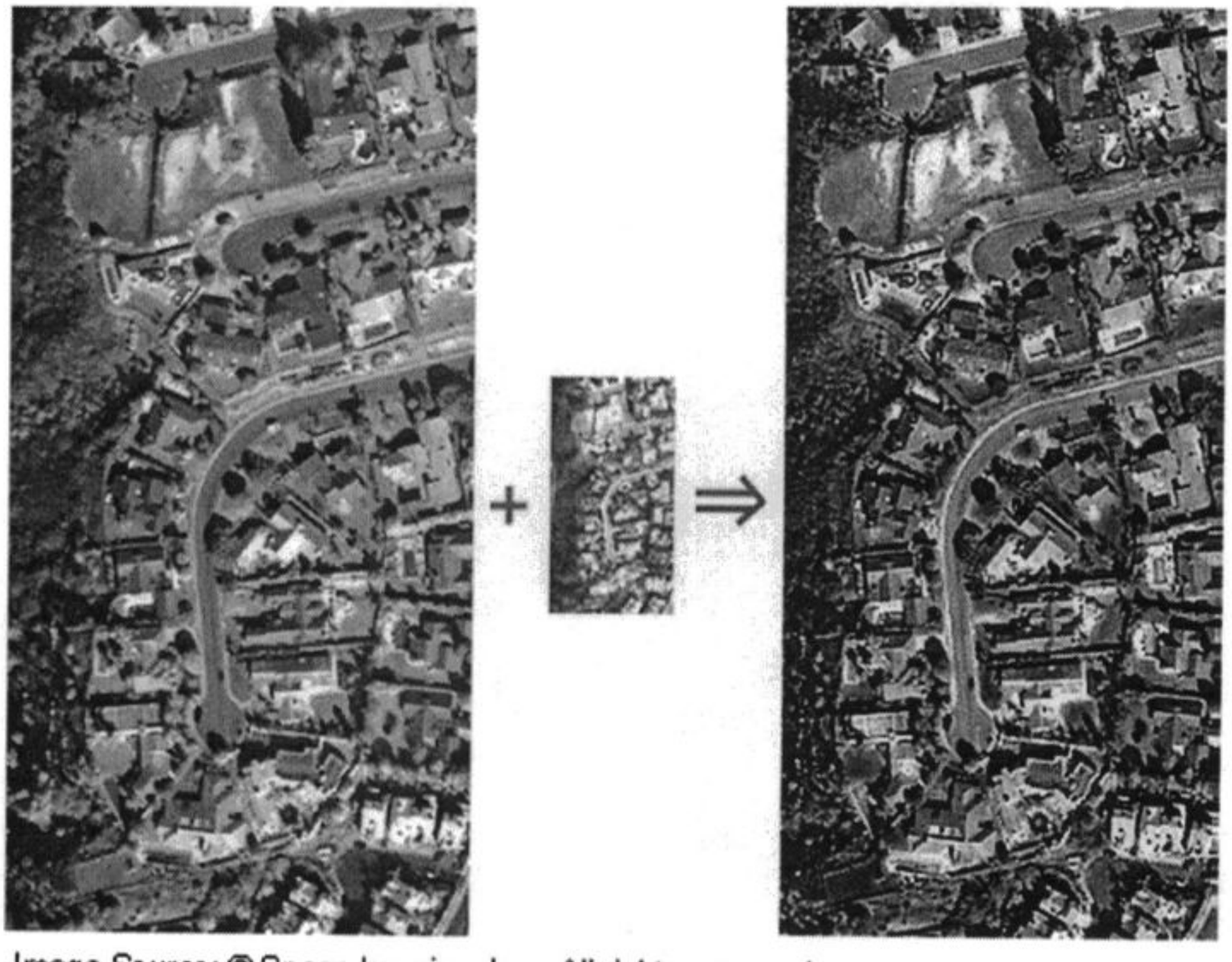


Image Source: © Space Imaging, Inc., All rights reserved.

Figure 1 : Le pansharpening

De pouvoir déduire des pixels dans l'image spectrale à partir d'une seule bande permet de réduire considérablement les coûts de fabrication des satellites. Les travaux de Gonzalez sont un exemple des recherches qui se font en ce sens [27]. Dans la même veine, ce genre de fusion des images pourrait aussi servir dans des systèmes où un zoom progressif est requis, par exemple dans des systèmes d'imagerie médicale ou militaire.

La fusion de représentations est aussi étroitement liée aux bases de données, car son rôle est d'uniformiser l'accès aux données, rôle qui est comparable à l'image qu'on se fait d'une base de données. On considère la fusion de représentations comme étant une couche supplémentaire ajoutée à un système de bases de données pour le rendre plus performant lorsque les données à traiter sont de nature spatiale ou géographique. Certains systèmes de bases de données, nommés systèmes de bases de données spatiales, intègrent cette couche directement.

Les chercheurs s'intéressent non seulement à rendre l'accès géométrique aux données plus performant [1] mais aussi à rendre les recherches de proximité [44] plus efficaces.

De manière générale, la fusion de représentations permet aux applications en télédétection de tirer parti de plusieurs sources différentes. En télédétection, on manipule des données provenant de capteurs différents, et le plus grand défi est d'arriver à trouver un terrain commun pour parvenir à combiner les propriétés de chaque source.

Le but de la fusion de données est de pouvoir condenser de l'information provenant de sources différentes de manière à obtenir une information de plus grande qualité. La fusion de représentations, le sujet de ce mémoire, s'intéresse plus particulièrement à la génération d'une seule représentation à partir de plusieurs représentations différentes. Les représentations qu'on fait des données varient d'une source à l'autre et peuvent être de type matriciel, ponctuel ou vectoriel. La représentation générée apporte des caractéristiques qui conviennent mieux aux algorithmes pour lesquels elle a été conçue. Par exemple, plusieurs algorithmes connus fonctionnent uniquement sur des représentations matricielles, ce qui nécessite soit la conversion indépendante de chaque source utilisée ou la fusion des représentations de chaque source.

La fusion de données en général nécessite des choix et des compromis, ce qui implique certaines pertes par rapport au signal d'origine. C'est au scientifique de développer une technique de fusion qui augmente le plus possible la qualité de l'information recherchée, et minimise le bruit et autres données indésirables. Dans certains cas, la fusion de données s'apparente à la réduction de la dimensionnalité en ce sens qu'elle sacrifie (ou filtre) certaines informations (le bruit) pour en faire ressortir d'autres, plus importantes (des objets).

Ce document est organisé de la façon suivante. Tout d'abord, ce mémoire fournit quelques bases théoriques sur la fusion de représentations et le rôle joué par les index spatiaux. Nous verrons comment fonctionnent les différents logiciels de géomatique, et pourquoi ils ne sont pas vraiment conçus pour le prototypage d'algorithmes en télédétection. Ensuite, nous présenterons la plateforme logicielle qui a servi de support à nos tests et nous détaillerons certains aspects de sa conception. Finalement, les résultats sont présentés et analysés. Au total, six méthodes ont été évaluées et comparées : cinq méthodes existantes et une nouvelle méthode, les tables d'enchaînement, qui est détaillée un peu plus loin dans ce document.

1.1 Les différents types de données en télédétection

Les données utilisées en télédétection peuvent être regroupées en trois formes de représentations différentes: les données matricielles, les données ponctuelles et les données vectorielles. Ces trois formes seront décrites en détail dans les pages qui suivent.

Ces trois formes de données issues des technologies modernes apportent toutes leurs avantages et leurs inconvénients. En tant que chercheurs dans le domaine de la télédétection, nous aimerions être capables de pouvoir tirer parti des avantages que présente chacune de ces technologies pour rendre nos algorithmes plus performants et plus efficaces.

Le problème, c'est que les algorithmes conçus pour effectuer la détection d'objets s'adaptent mal à des sources de données différentes des images traditionnelles :

1. Les technologies sont différentes et sont présentées sous des formats de données différents. Les représentations, la précision des variables et la structure des données varient.
2. Les données, les images, sont sauvegardées sous des résolutions différentes, des repères géographiques différents.
3. Les algorithmes sont souvent construits pour répondre à un besoin particulier, qui se limite à un format et un cadre d'opération précis.

En télédétection, le terme « image » est utilisé. Il ne faut pas confondre une image de télédétection avec une image simple. L'image de télédétection comporte généralement divers canaux qui peuvent présenter différentes caractéristiques et il est important de comprendre que l'image est en réalité générée à partir de ces canaux.

Les pages qui suivent décrivent en détail les trois formes communes de représentations en télédétection.

1.1.1 Données matricielles

Les données matricielles s'inscrivent aisément sur des matrices parce qu'elles ont des dimensions horizontales et verticales qui sont définies. Une autre propriété importante de ce type de données est que la résolution (l'espacement entre les points) pour une même carte est définie, régulière et précise.

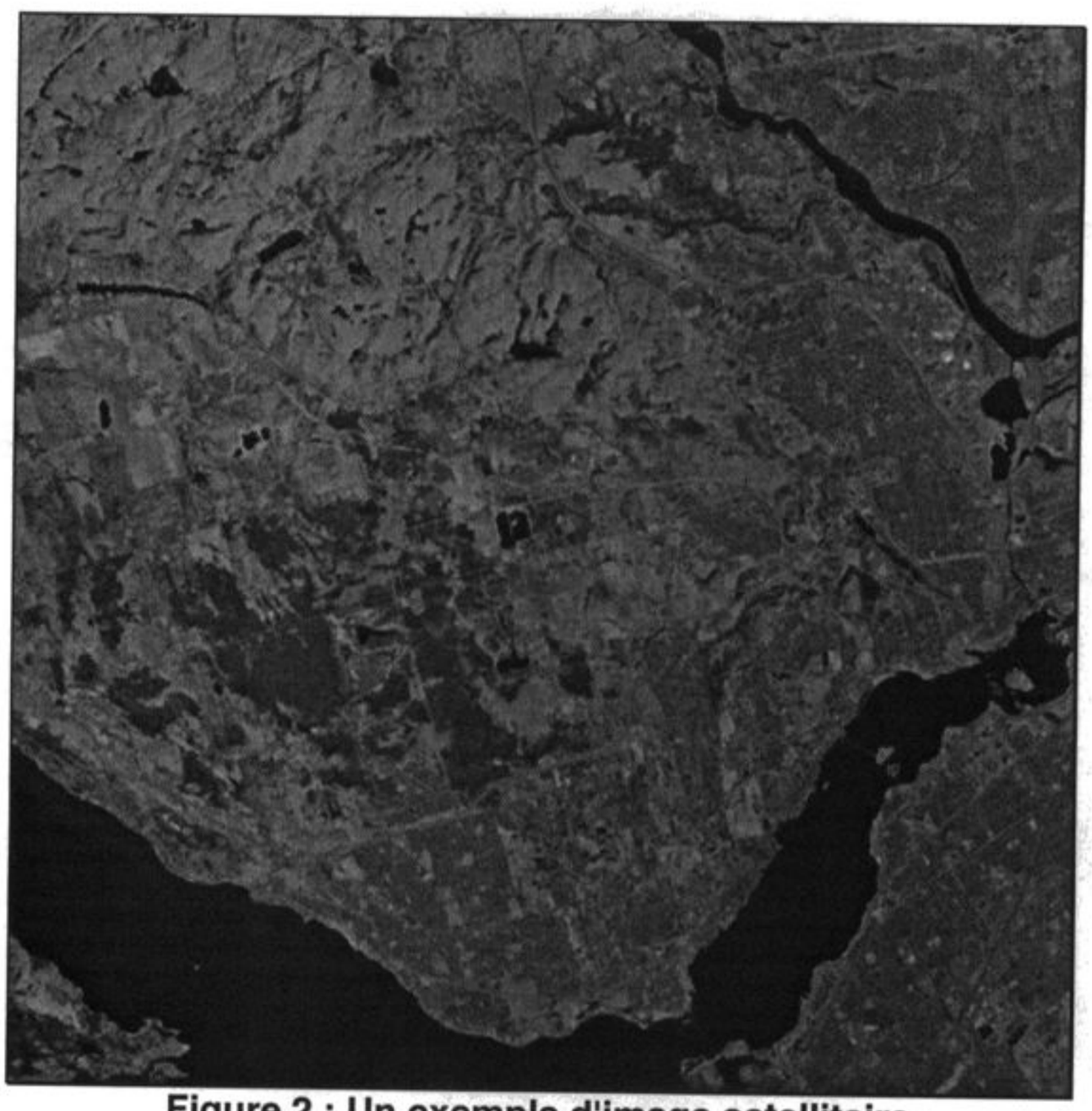


Figure 2 : Un exemple d'image satellitaire

L'image la plus courante est l'image satellitaire (Figure 2). Ce type d'image est généralement constitué de plusieurs canaux provenant de différents capteurs dont le satellite est muni. Des opérations de géorectification et de géoréférencement sont nécessaires avant l'utilisation de l'image par un algorithme. De plus, les

capteurs peuvent présenter différences au niveau de la résolution et de la précision, rendant le travail encore plus complexe.

Il existe aussi les images provenant de caméras photographiques sophistiquées placées sous les avions (Figure 3). Ces données sont généralement plus simples lorsqu'il s'agit d'images couleur normales, mais il est possible d'utiliser d'autres capteurs et cela dépend de l'application visée. De manière générale, il existe un canal par capteur.

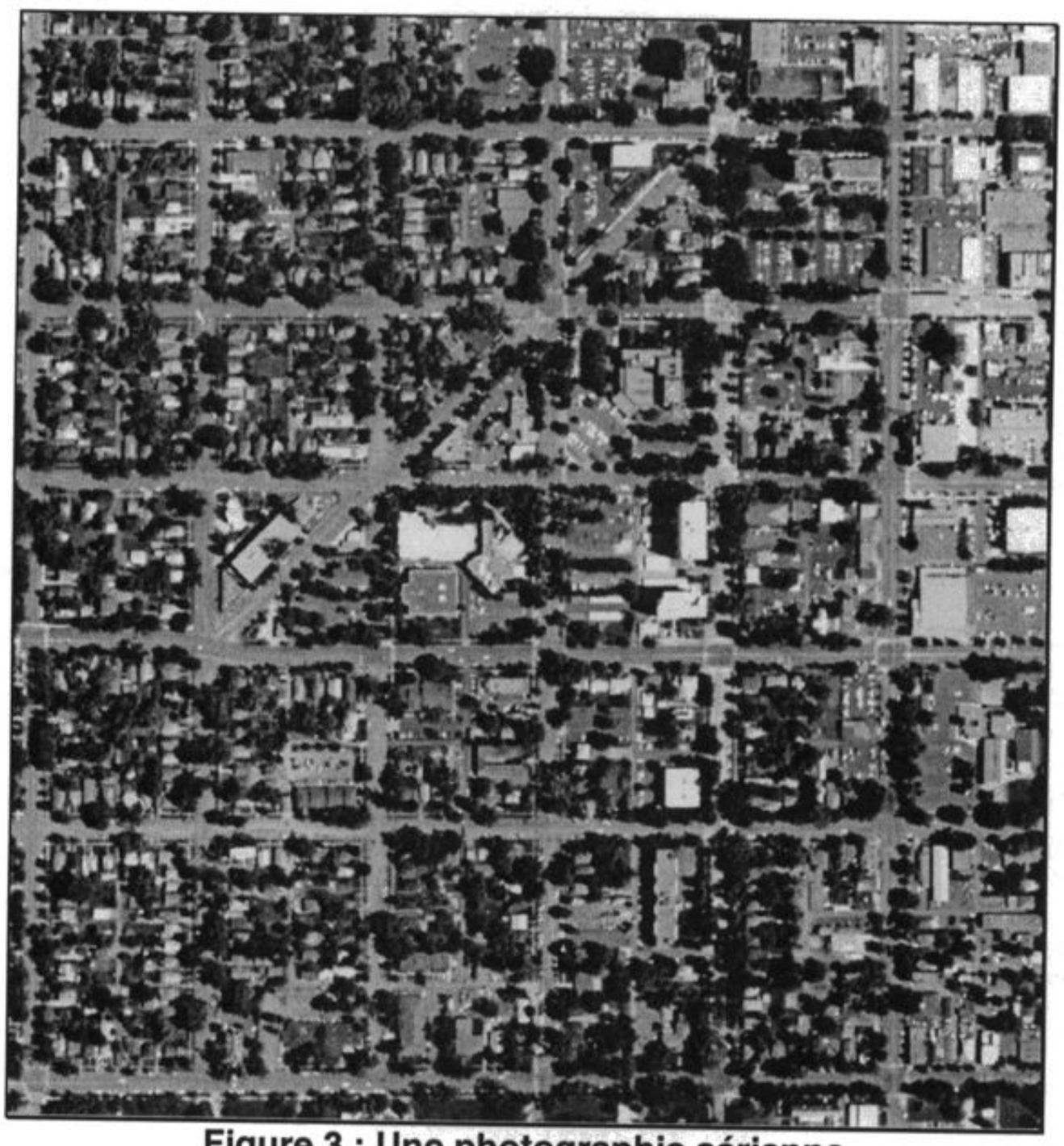


Figure 3 : Une photographie aérienne

1.1.2 Données vectorielles

Dans un contexte de télédétection, les données vectorielles font référence à de l'information qui est notée sous forme vectorielle. La forme vectorielle est celle qui décrit la forme et le contenu d'un objet. De manière pratique, cette méthode de description des données est utilisée pour stocker les informations cartographiques (Figure) ou encore pour améliorer l'efficacité de traitement de données éparpillées géographiquement.

Figure 4 : Un exemple de données vectorielles

Puisqu'elles sont définies à la fois par leur forme et leur contenu, les données vectorielles peuvent facilement inclure des données matricielles dans un contexte qui ne s'y limite pas (voir Figure). Il n'est plus question de résolution, mais seulement de points de référence et de tailles. Les données vectorielles sont probablement les plus difficiles à manipuler parmi les trois catégories.

1.1.3 Données ponctuelles

Les données ponctuelles sont en quelque sorte une fusion des concepts rencontrés dans les deux catégories précédentes : elles désignent les données qui sont fournies sans structure particulière, uniquement une position et une valeur. L'imagerie laser LIDAR (Figure) est un bon exemple de technologie utilisant des données ponctuelles comme méthode de représentation du contenu.

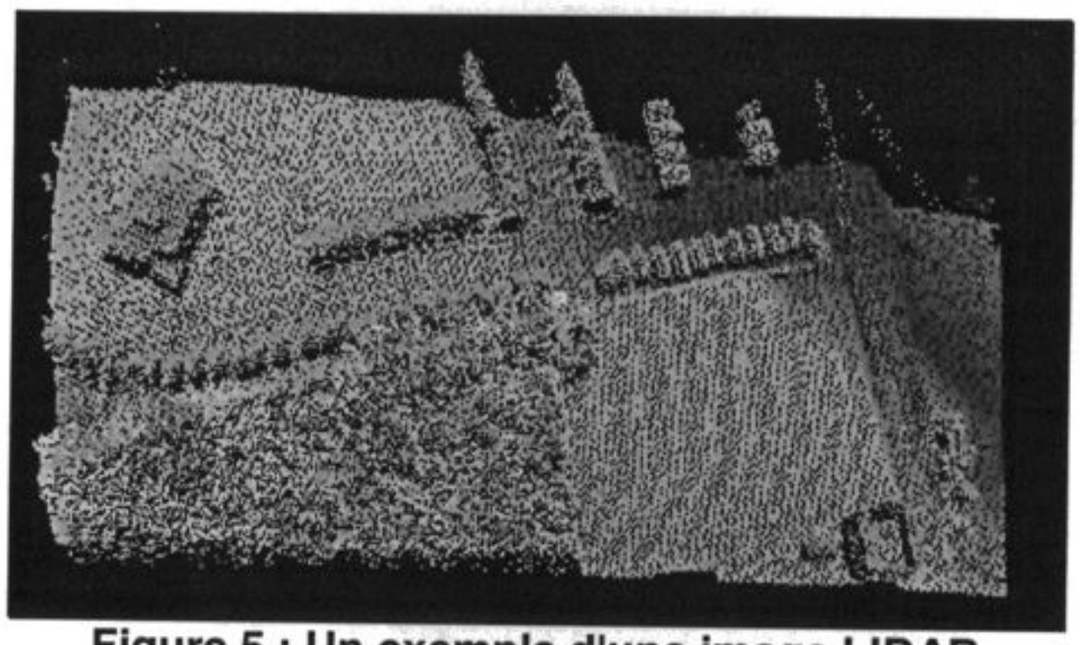


Figure 5 : Un exemple d'une image LIDAR

Les données ponctuelles ne possèdent aucune structure et ne garantissent pas que la position des points présentés soit parfaitement alignée sur des grilles à résolution fixe. L'espacement entre les points peut être aléatoire et la résolution est calculée comme une moyenne de ces espacements au lieu d'être une valeur fixe et précise. Intrinsèquement, une structure par points permet la manipulation en temps réel d'un flux de données.

1.2 Prototypage rapide en télédétection : un problème fonctionnel

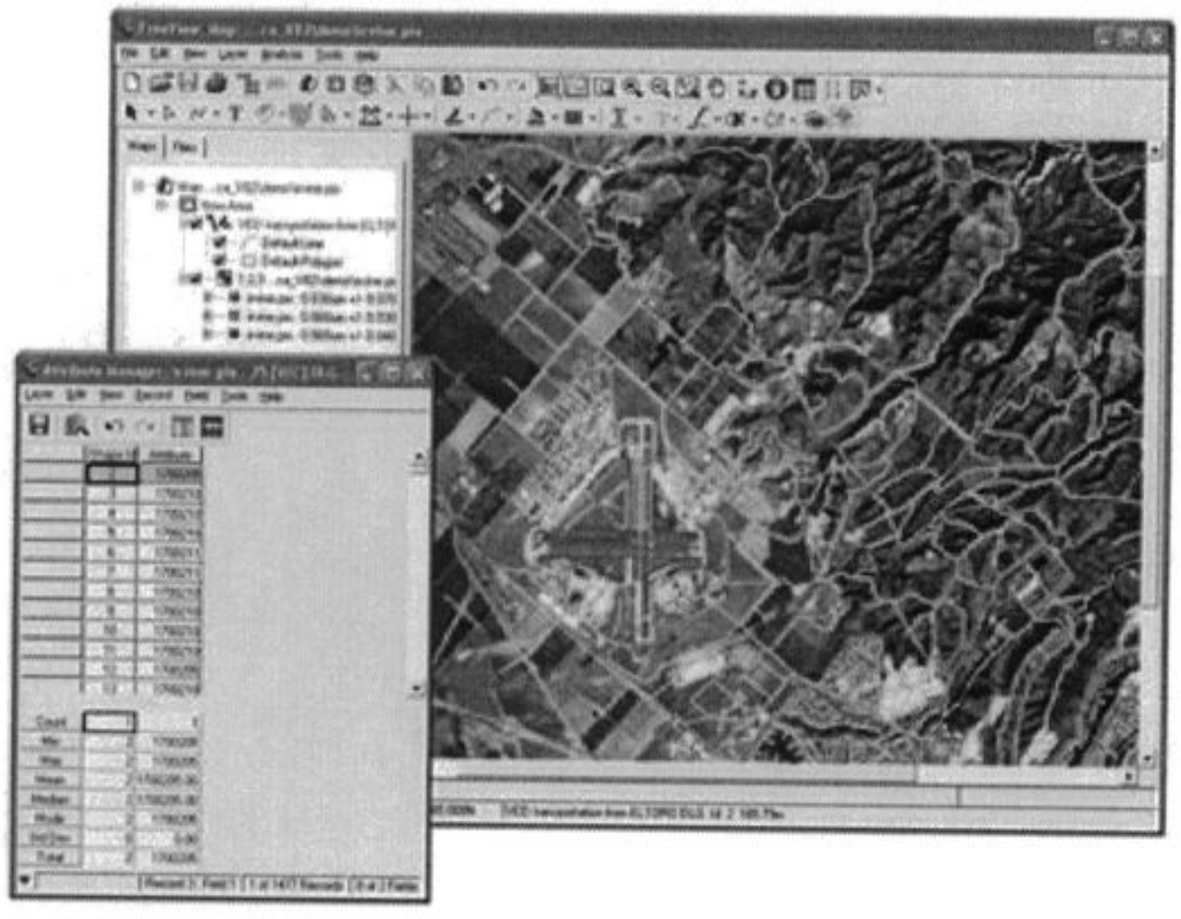
Le développement d'applications en télédétection nous amène à travailler avec plusieurs sources de données différentes. Les sources utilisées se distinguent par plusieurs caractéristiques, dont voici les plus importantes : la résolution, la taille, les capteurs, mais surtout et avant tout la représentation spatiale des données.

Un objectif commun est de pouvoir implémenter des algorithmes de télédétection rapidement en vue de découvrir une méthode qui puisse améliorer les techniques connues. Il existe peu d'options : l'utilisation d'un logiciel de géomatique existant, ou alors la création de programmes qui permettront de tester les algorithmes intéressants. La plupart des chercheurs commencent par développer des programmes simples fonctionnant sur des images fixes, puis plus la complexité augmente, plus il y a des chances qu'ils décident d'utiliser un logiciel commercial. C'est un comportement raisonnable : les différents types de représentation de données sont au coeur du problème auquel est confronté le chercheur, car ils évoquent des incompatibilités sérieuses entre les algorithmes utilisés. Non seulement faut-il que les algorithmes soient modifiés pour interpréter correctement la nature des données d'entrée, mais en plus il faut les adapter pour qu'ils puissent reconnaître les résultats d'autres algorithmes de leur chaîne de traitement. En passant à une plateforme commerciale, la complexité due à la représentation des données est réduite partiellement, car le logiciel s'en charge. Partiellement, car le chargement des données ne se fait pas toujours automatiquement : il faut donner plusieurs paramètres sans quoi les données risquent d'être mal interprétées. Tous les logiciels font, dans une certaine mesure, abstraction de la fusion de représentations, ce qui peut être vu comme une bonne chose pour l'utilisateur en général, mais, comme nous le verrons, ça ne l'est pas nécessairement pour le chercheur qui prototype de nouveaux algorithmes.

Dans le domaine commercial, il existe des plateformes importantes pour la télédétection. PCI Geomatics, ERDAS Imagine, IDRISI et ESRI ArcGIS sont des bons exemples. Nous avons eu la possibilité d'étudier et de comparer deux de ces plateformes que nous vous présentons dans les pages qui suivent. La comparaison a été faite sur trois aspects : l'importation des données, la fusion de représentations et les possibilités en termes de prototypage.

1.2.1 PCI Geomatica 9 et 10 (Focus)

PCI Geomatica est un logiciel spécialisé dans le traitement des images satellitaires. Il est capable d'images ponctuelles ou vectorielles, mais seulement en sacrifiant certains aspects de performance.



Tout d'abord, le chercheur doit convertir (importer) toutes ses données dans un format interne à PCI. Ce format (PIX) est un format matriciel. Par exemple, si on importe des données ponctuelles, elles seront rastérisées. On comprend déjà mieux pourquoi on dit que PCI Geomatica est spécialisé dans les images satellitaires.

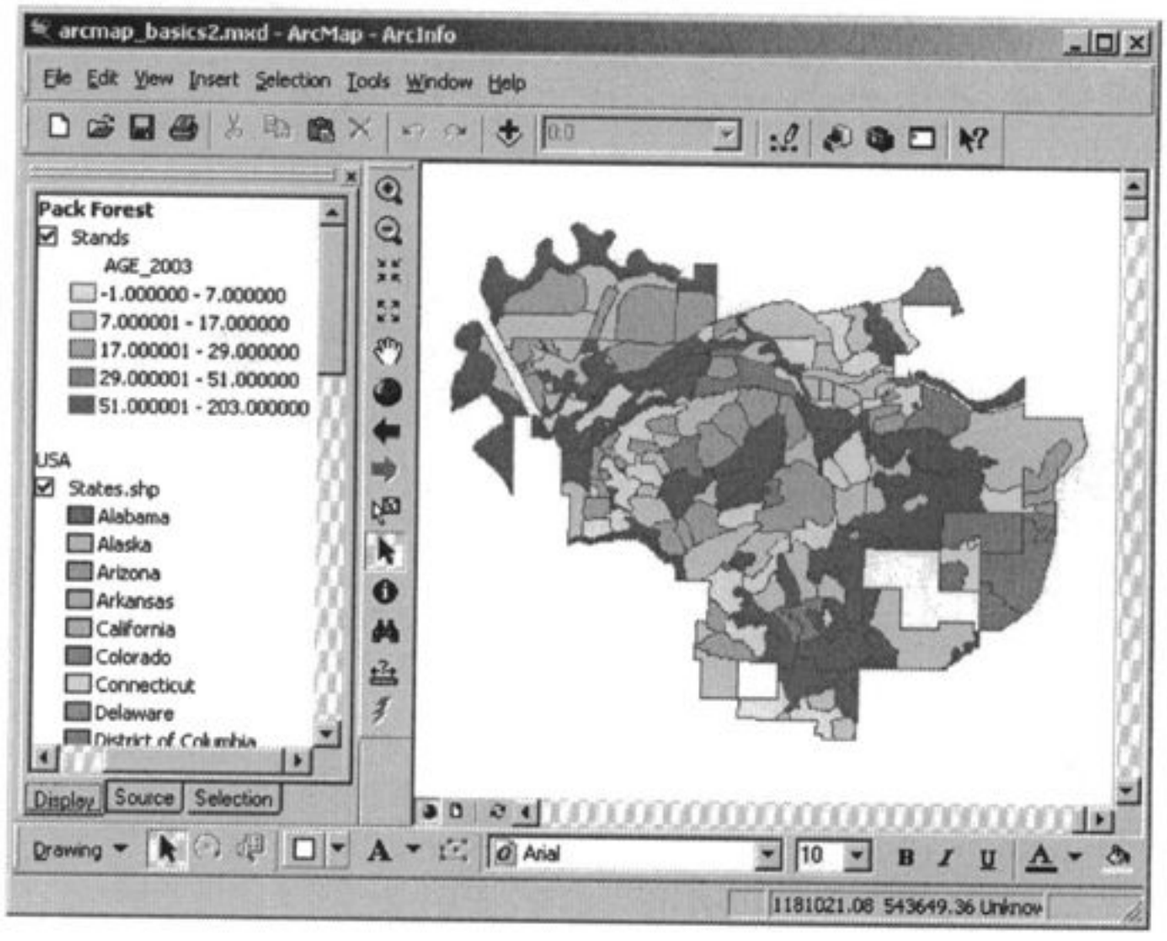
La fusion de données en Géomatica est surtout d'ordre graphique. Il est possible d'afficher plusieurs représentations dans une même vue. Pour le traitement des données cependant, c'est autre chose : il est impossible d'effectuer des opérations sur des couches de données qui soient incompatibles. Des couches incompatibles sont des données de résolutions différentes ou alors des données qui

ne représentent pas exactement le même terrain (*different map extents*). Pour effectuer un traitement sur des couches spatiales disparates, il faut d'abord choisir une référence, et convertir toutes les couches sur cette référence. Nous obtenons alors un nouveau fichier. À titre d'exemple, une image basse résolution ramenée à une référence haute résolution produira un gros fichier puisque les pixels de l'image basse résolution seront répliqués de multiples fois de manière à obtenir une (fausse) image haute-résolution.

Au niveau du prototypage, Geomatica offre la possibilité d'écrire des scripts. Les scripts sont écrits dans le langage propre à la plateforme. Il existe la possibilité d'écrire ses propres plugins en C++, mais il faut acheter la version entreprise du logiciel, qui est coûteuse.

1.2.2 ESRI ArcGIS (ArcMap)

ArcGIS est un logiciel spécialisé dans l'affichage et la présentation d'images vectorielles. Il est capable d'images matricielles ou ponctuelles tout comme Geomatica, mais avec certaines pénalités. Contrairement à Géomatica, la force de la plateforme se situe au niveau de la qualité et le professionnalisme de l'affichage, au détriment de la puissance de traitement.



Au niveau de la gestion des données, ArcGIS est beaucoup plus souple. Il utilise une base de données pour stocker l'information dans l'un ou l'autre des formats internes proposés. ArcGIS possède trois formats internes : matriciel, ponctuel et vectoriel (Map, Point, Vector). C'est un avantage important face à Geomatica qui fonctionne sur un format unique (PIX).

La fusion des représentations est purement graphique. Les données sont laissées dans leur format d'origine et sont séparées en fichiers distincts. L'affichage se fait en douceur grâce à un mécanisme d'affichage progressif. Le traitement des données par des algorithmes ne peut être réalisé que sur un fichier à la fois. Pour cette raison, si le chercheur veut effectuer un traitement multi-résolution ou multi-format, il doit encore une fois forcer la conversion des données d'un format à l'autre pour regrouper le tout dans un seul fichier à plusieurs couches. Là aussi, beaucoup de pertes sont subies.

Pour ce qui est du prototypage, ArcGIS fonctionne uniquement à l'aide de scripts VisualBasic ou Python. L'avantage est que ces langages de scripts sont mieux connus. L'inconvénient est qu'il n'y a pas de support pour des plugins. L'intérêt principal des plugins est que l'algorithme est davantage portable, puisque le code peut être réutilisé dans d'autres projets, contrairement à un script qui devient lié à la plateforme sur laquelle il peut être exécuté.

1.2.3 Comparaison des plateformes

Plateforme	Format Interne des données	Fusion de représentations	Prototypage rapide
PCI Geomatica	Matriciel pur	Graphique seulement	Scripts et Plugins
ESRI ArcGIS	Multiple	Graphique seulement	Scripts seulement

1.3 La fusion de représentations appliquée à la télédétection : un problème technique

Les plateformes de traitement en télédétection n'offrent malheureusement pas la possibilité d'utiliser la fusion de représentations. Nous l'avons vu précédemment, la fusion utilisée est purement visuelle, les algorithmes eux n'ont d'autre choix que de travailler dans un contexte bien déterminé. À cause de ces différences importantes entre les algorithmes et les technologies sur lesquelles on aimerait les utiliser, nous sommes contraints de devoir adapter chaque algorithme à chaque source, de manière indépendante et isolée, afin de pouvoir les exécuter dans un ordre prévu à l'avance. De plus, il arrive souvent d'être simplement incapable d'enchaîner deux algorithmes sans avoir à en modifier un intégralement. La figure suivante illustre le genre de problèmes que l'on rencontre couramment en télédétection :

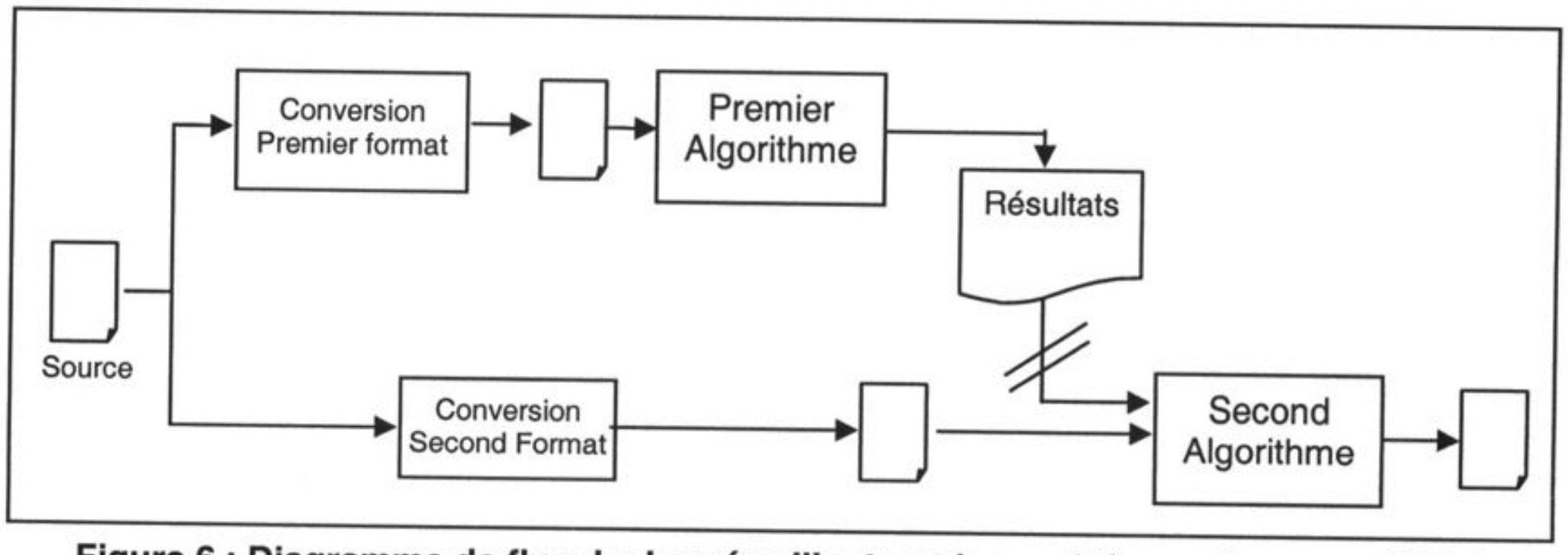


Figure 6 : Diagramme de flux de données illustrant les problèmes de compatibilité

Ce sont ces problèmes de compatibilité qui nous motivent à utiliser deux outils importants : le génie logiciel pour concevoir une plateforme réutilisable et la fusion de données pour rendre une ou plusieurs sources compatibles avec les algorithmes traditionnels, sans avoir à les modifier.

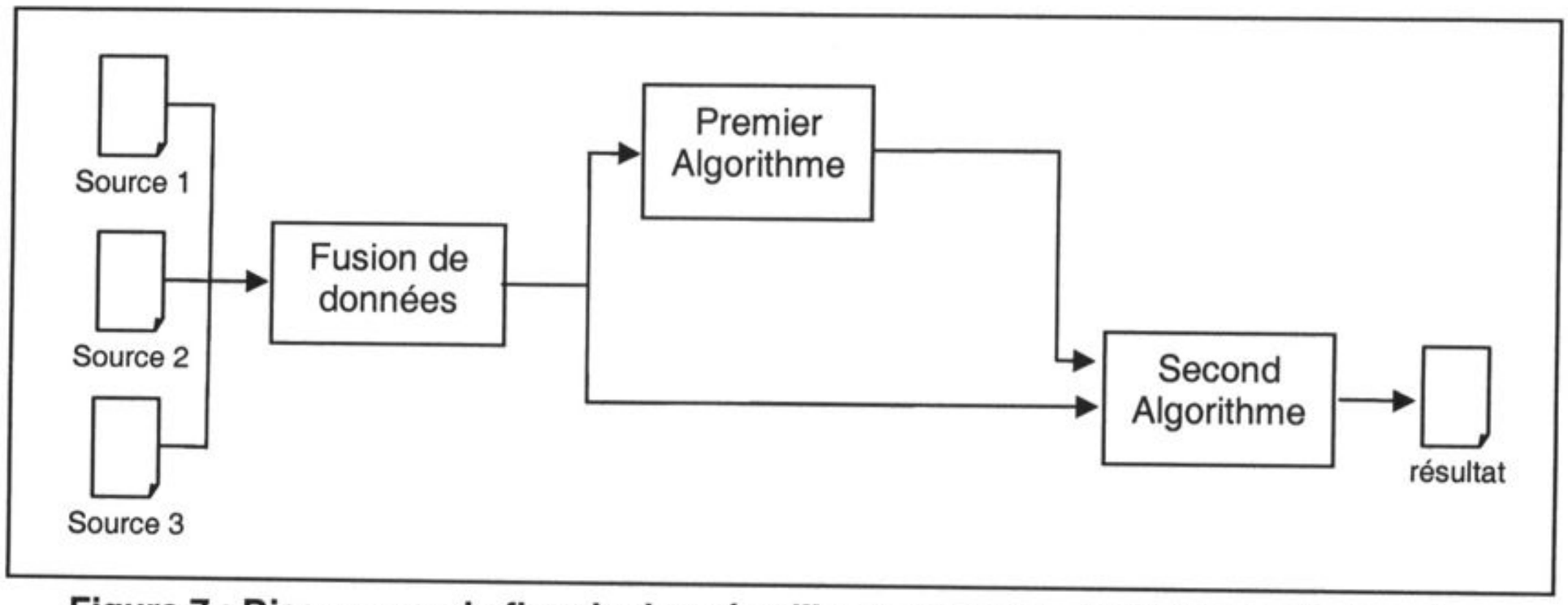


Figure 7 : Diagramme de flux de données illustrant le rôle de la fusion de données

D'une part, la plateforme FeatureEx-RA permettra la réutilisation des modules algorithmiques et l'imposition d'une certaine structure de travail. D'autre part, la fusion de représentations permettra l'abstraction des sources de données utilisées en rendant l'accès transparent pour l'algorithme ou l'utilisateur.

Nous aimerions qu'au terme de ce mémoire nous puissions créer des algorithmes qui travaillent sur des sources de données multiples de manière transparente. Idéalement, les algorithmes seraient génériques et enchaînables comme bon nous semble.

Sans perdre de vue l'importance de la plateforme FeatureEx-RA, l'objectif de ce mémoire est d'implanter différentes techniques d'indexage spatial au sein de Feature Ex-RA et d'évaluer leurs performances dans la fusion de représentations.

La section suivante présente une introduction à la fusion de données et à ce qu'on entend par représentation et la fusion des représentations. Plusieurs techniques de fusion de représentations sont détaillées et cinq d'entre elles seront implémentées, analysées et comparées au terme de ce mémoire. De plus, une nouvelle méthode, la méthode des tables d'enchaînements, sera présentée avec les autres.

De manière plus générale, PCI Geomatica est un logiciel destiné au traitement des images satellitaires, et ArcGIS est destiné à la réalisation de cartes.

1.4 La situation idéale pour un chercheur en télédétection

Le chercheur en télédétection aimerait avant tout pouvoir développer ses algorithmes avec le moins d'entraves possible. S'il décide de tout développer lui-même, il fait face à une complexité grandissante. S'il décide d'opter pour une plateforme commerciale, il perd en flexibilité, plus particulièrement au niveau de la fusion de représentations. Une situation idéale serait de pouvoir avoir le meilleur des deux mondes : une plateforme à la fois flexible et simple à utiliser.

Cette réflexion est à l'origine de notre décision de joindre à ce mémoire nos travaux sur une plateforme de prototypage rapide que nous avons baptisée Feature Ex-RA. L'objectif principal de cette plateforme est de permettre au chercheur d'effectuer des tests rapidement, avec le moins de tracas possible. De manière générale, le processus de prototypage se résume comme suit :

1) Le chercheur importe les données dans une base de données (comme ArcGIS)
2) Le chercheur implémente son algorithme sous la forme d'un plugin
3) Le plugin est pris en charge automatiquement par la plateforme
4) Le chercheur choisit l'algorithme de fusion de représentations de son choix.
5) Le chercheur exécute son algorithme sur les données qui l'intéressent. (le type de source n'importe plus et est pris en charge par la fusion de représentations)
6) Le chercheur recueille les résultats, les évalue et poursuit ses tests au besoin.

Tous les résultats présentés dans ce mémoire sont obtenus au sein de cette plateforme qui sera décrite plus en détail un peu plus loin.

2 Objectifs du mémoire

Globalement, l'objectif de ce mémoire est d'implanter, à l'intérieur d'une plateforme permettant le prototypage rapide d'algorithmes, différentes techniques d'indexage spatial et évaluer leurs performances dans la fusion de représentations.

D'une part, il y a les index spatiaux, et d'autre part il y a le moteur de fusion et la plateforme dans laquelle il se trouve. L'un ne peut fonctionner sans l'autre, et c'est la raison pour laquelle nous présentons dans ce mémoire d'abord la plateforme Feature Ex-RA qui a été développée pour réaliser les tests sur les index spatiaux.

Les index spatiaux ont été réalisés séparément et évalués au sein de la plateforme.

M. Weihe a évoqué le manque à gagner [42] au niveau de l'interopérabilité des algorithmes développés dans la communauté scientifique et le besoin de l'élaboration d'une plateforme standard de travail. La plateforme a été réalisée avec un souci de réutilisation et de flexibilité qui suit cette école de pensée.

Les techniques d'indexage qui ont été choisies pour implantation dans la plateforme sont :

- Grille
- Grille à baquets
- Arbre R
- Arbre quaternaire matriciel
- Arbre k-d

Les raisons pour lesquelles nous nous sommes limités à ces techniques sont simples : il en existe trop (il fallait choisir), elles sont facilement adaptables à des espaces multidimensions (elles sont pertinentes) et elles sont retenues et reconnues

par la communauté scientifique (elles ont fait leurs preuves). Ce mémoire présentera les performances de ces différentes méthodes d'indexage au sein de la plateforme Feature Ex-RA.

Une nouvelle technique d'indexage, la méthode par tables d'enchaînement, a été créée pour répondre à certaines lacunes de la méthode de la grille à baquets. Elle sera présentée au même titre que les cinq autres méthodes mentionnées ci-dessus.

3 État de l'art

3.1 La fusion de données

Bien que plusieurs définitions de la fusion de données aient été proposées dans la littérature, celle de Lucien Wald [41] est certainement la plus communément admise. Les définitions proposées avant lui ne tenaient pas compte de la nature générale du domaine de la fusion de données et imposaient des limites quant aux longueurs d'onde utilisées dans les capteurs, ou alors étaient trop orientées vers des méthodes particulières.

Lucien Wald a défini la fusion de données comme étant "un cadre formel dans lequel s'expriment les moyens et techniques permettant l'alliance des données provenant de sources diverses. Elle vise à l'obtention d'information de plus grande qualité ; la définition exacte de « plus grande qualité » dépendra de l'application."

Bien au-delà de cette définition, nos recherches ont permis de construire une base de termes et de définitions solides à utiliser au cours de la rédaction du mémoire. La table qui suit présente des explications sur plusieurs termes importants dans un contexte de télédétection.

Table 1 : définitions et terminologie en télédétection

Terme français	***Définition***
Capteur	Dispositif influencé par une grandeur physique et fournissant un signal qui représente la valeur de celle-ci.
Mesure	La valeur d'un signal reçu à la sortie d'un capteur [41]. Selon le cas, on peut aussi parler de signal (1-D) ou d'image (2-D).
Échantillon	L'ensemble des mesures recueillies par un même équipement pour une même cible à un instant donné est appelé un échantillon. Dans le cas des images, il est aussi possible d'utiliser le terme pixel. De manière générale, on utilise aussi les termes vecteur d'état ou vecteur d'attributs.
Attribut	Une propriété d'un échantillon. Dans certains cas, on peut parler de variable d'état d'un échantillon.
Représentation / Vue	Présentation des attributs selon diverses contraintes. Ces contraintes peuvent être spatiales, temporelles ou basées sur les valeurs des attributs. Une représentation peut concerner un ou plusieurs objets et peut aussi être vide, tout dépendant des contraintes imposées.
Règles	Elles sont des équations ou des expressions logiques qui définissent la syntaxe liant des éléments entre eux.
Décisions	Les décisions découlent de l'application des règles et provoquent des réactions du système aux stimuli.
Équilibrage	Opération qui consiste à ramener des données disparates sur une même base spatiale et temporelle. On parle parfois d'harmonisation ou de conditionnement. [5, 25]
Association	Effectuée après l'équilibrage, cette opération consiste à regrouper des vues / représentations fortement corrélées entre elles. Par exemple, les vues LIDAR et satellite équilibrées d'un même lieu géographique peuvent être associées ou concaténées pour ne former qu'une seule vue présentant l'ensemble des attributs contenus dans les deux vues séparées.
Classification	Opération qui consiste à grouper formellement des éléments de configuration, des incidents, des problèmes ou des changements, en fonction de leur type. [30] En télédétection, ces groupements sont effectués sur la base des attributs des échantillons recueillis en fonction de leur classe d'appartenance.
Classe / Étiquette / Catégorie / Taxon	Noms donnés aux regroupements produits par une opération de classification.

Dans [5], les auteurs répertorient plusieurs spécialisations de la fusion de données qu'ils ont rencontrées dans la littérature. La table ci-dessous présente quelques termes mentionnés dans leur rapport, sans pour autant s'y limiter et auxquels a été ajoutée une courte présentation du contexte dans lequel ces termes ont été rencontrés dans plusieurs autres articles.

Table 2 : variétés de fusion de données rencontrées dans la littérature

Terme français	***Domaine***	***Contexte***
fusion de position	Militaire	position d'une cible sur plusieurs systèmes géographiques. Augmentation de la précision de position par fusion de plusieurs mesures de position.
fusion de classe	Militaire	classification de cibles. Augmentation de la certitude de la classe d'appartenance d'une cible (class identity) par fusion des résultats de plusieurs méthodes de classification sur plusieurs échantillons
fusion multicapteurs	Général	Fusion des mesures de plusieurs capteurs pour une même cible.
fusion de mesures	Général	Fusion des mesures obtenues pour obtenir d'autres mesures plus intéressantes.
fusion d'image	Imagerie	Fusion de plusieurs images pour révéler des détails normalement invisibles ou difficiles à voir dans les images séparées.
fusion de pixel	Imagerie	Fusion d'image réalisée sur la base des pixels. (pixel par pixel)
fusion prouvée[1]	Mathématiques	Fusion fondée sur la théorie des preuves
fusion en logique floue	Statistiques	Fusion utilisant des techniques de logique floue.
fusion géométrique, fusion spatiale, fusion de représentations	Géomatique	Ces termes font référence au même concept : la fusion de représentations.

[1] Le terme français utilisé est une suggestion de l'auteur pour traduire "evidential fusion". Aucune traduction formelle n'a été trouvée pour cette catégorie de fusion de données.

Plus spécifiquement à la télédétection, on rencontre surtout les termes *fusion de données*, *fusion de capteurs*, ou *fusion d'images*. Chacun de ces termes est employé dans un contexte spécifique. Par exemple :

Table 3 : trois catégories de fusion fréquentes en télédétection

fusion de données	concerne des applications où la fusion est de bas niveau : on traite des données brutes qui n'ont pas nécessairement encore acquis de forme. (d*ata fusion*)
fusion de capteurs	concerne des applications où la technique de fusion vise à regrouper les données récoltées de plusieurs capteurs différents. (*multi-sensor fusion*)
fusion d'images	concerne des applications où des images différentes sont fusionnées en une nouvelle image améliorée. (*image fusion*)

De manière plus générale, toutes ces variétés de techniques de fusion de données peuvent être classées en trois formes universelles :

- La fusion des attributs
- La fusion des analyses
- La fusion des représentations ou des vues

3.1.1 Fusion d'attributs

La fusion des attributs regroupe toutes les opérations qui visent à fusionner des attributs pour créer de nouveaux attributs. Ces nouveaux attributs présentent alors des indices plus clairs quant à la nature de l'échantillon analysé. Ce type de fusion est utilisé pour améliorer le résultat des analyses. Un attribut peut être n'importe quelle mesure.

Fusion de sources compatibles

David A. Fay présente un bon exemple de fusion de données sur les attributs dans son article "Multisensor & Spectral Image Fusion & Mining: From Neural Systems to Applications" [7]. Il utilise la plateforme ERDAS Imagine comme base de travail et il a développé des outils qui utilisent des bandes spectrales visibles et invisibles pour créer un mélange qui puisse faire ressortir les cibles des objets qui n'en sont pas. L'illustration suivante est tirée de son article et démontre le mécanisme de fusion des attributs qui lui permet d'obtenir une image couleur en sortie (Figure).

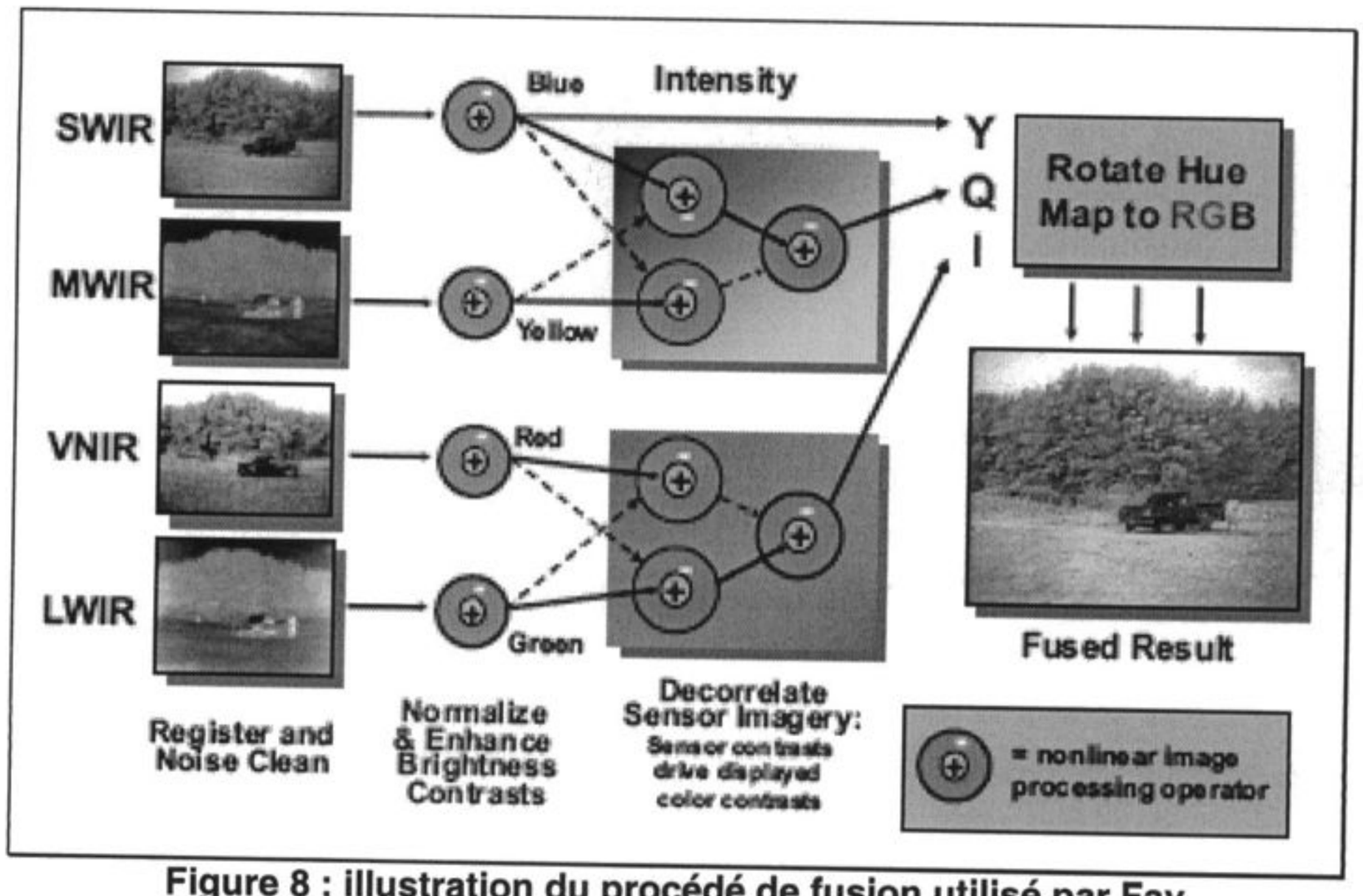


Figure 8 : illustration du procédé de fusion utilisé par Fay

Cet exemple d'application est intéressant en ce sens qu'il démontre une chaîne complète de fusion de données : d'abord, il utilise des bandes spectrales semblables (même dimension et même résolution) pour les fusionner en utilisant des algorithmes internes pour ensuite générer une représentation sous forme d'une image couleur.

Le travail de David Fay est conçu pour une application précise : la vision nocturne. Un appareil photographique est utilisé pour représenter une scène sous quatre bandes spectrales différentes. C'est équivalent à avoir quatre points de vues différents, car chaque bande opère dans un spectre électromagnétique distinct. Par exemple, nous savons bien que l'infrarouge est particulièrement utile pour reconnaître les sources de chaleur.

La fusion est réalisée en mélangeant les bandes deux par deux au moyen d'un algorithme non-linéaire. Les résultats préliminaires sont remélangés au moyen du même algorithme, ce qui donne en fin de compte trois bandes qui seront réarrangées pour être représentées comme des couleurs visibles par le système visuel humain. La méthode passe de quatre bandes distinctes à trois bandes interreliées : il y a eu perte d'information. Cette perte d'information est compensée par la mise en valeur des objets d'intérêt.

Fusion de sources incompatibles

Un autre projet intéressant que nous avons rencontré est celui de J. B. K. Kiema, qui a publié l'article "Wavelet Compression and Data Fusion: An Investigation into the Automatic Classification of Urban Environments using Colour Photography and Laser Scanning Data" [20]. M. Kiema relate les détails de ses travaux de détection automatisée des bâtiments et autres objets grâce à une fusion d'images aériennes et LIDAR (LIght Detection And Ranging). Il y relate l'importance de pouvoir utiliser les propriétés tridimensionnelles des données LIDAR pour améliorer

l'efficacité des algorithmes de détection et de classification. La figure (Figure) montre les images aériennes CIR (Colour InfraRed) et LIDAR :

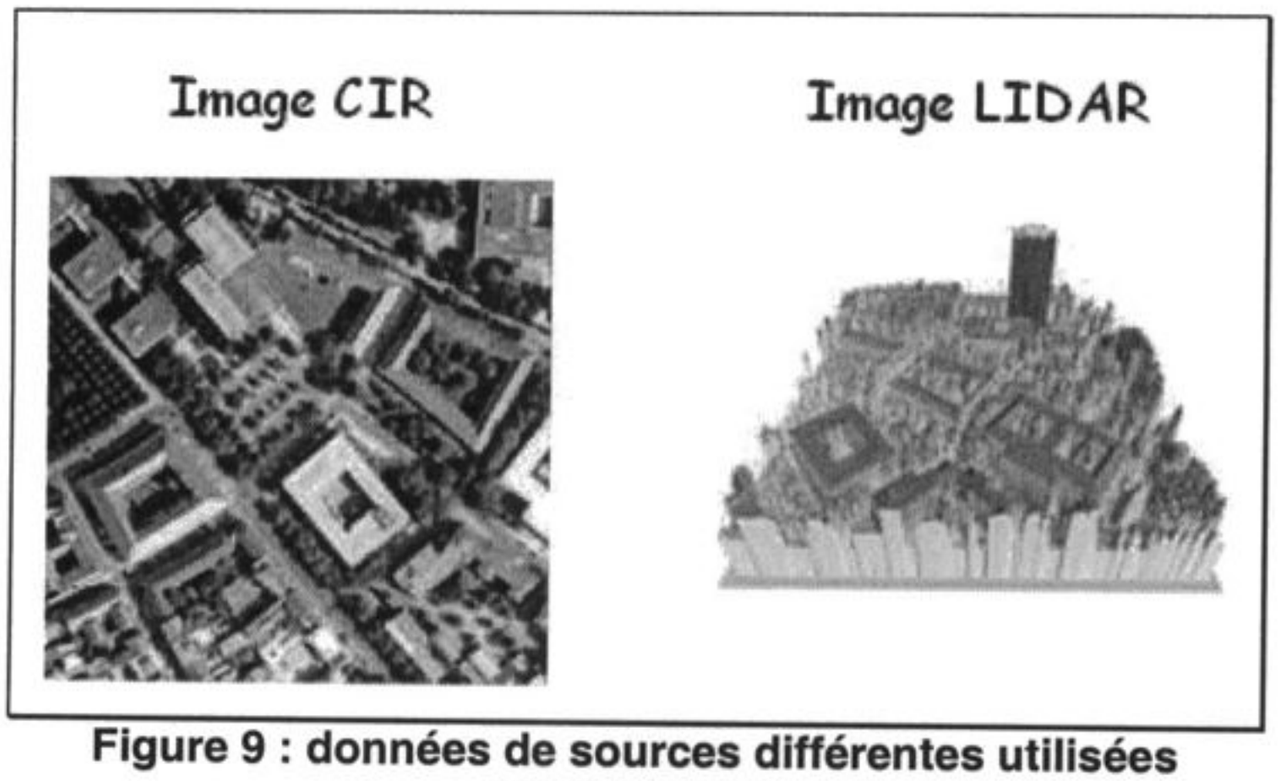


Figure 9 : données de sources différentes utilisées par M. Kiema

La méthode consiste à superposer l'image LIDAR sur l'image CIR et fusionner les deux images dans l'image CIR en ajoutant un canal à celle-ci. Ainsi, l'image CIR originale qui possédait quatre canaux (RGB.IR) est étendue à cinq canaux (RGB.IR.LIDAR), où ce cinquième canal correspond à un attribut de l'image LIDAR qui nous intéresse. Dans le travail de M. Kiema, c'est l'attribut « Altitude ».

Cette méthode est aussi appelée fusion de canal : les attributs conservés le sont intégralement, alors que les autres sont perdus. Ce type de fusion diffère de l'exemple précédent en ce sens que la méthode de M. Fay mélangeait certaines propriétés de toutes les bandes dans le résultat final, alors qu'en fusion de canal c'est tout un canal qui est fusionné à une autre image. Il n'y a pas de mélange.

Les attributs perdus lors de cette fusion sont : les attributs de position (easting, northing), l'attribut temporel et l'attribut d'intensité du signal. L'altitude sera utilisée pour aider les algorithmes de classification à distinguer un édifice d'une route par exemple. En ajoutant ce cinquième canal à l'image originale, l'algorithme verra l'édifice différemment de la route simplement à cause de la différence indiquée par l'attribut d'altitude.

Nous avons retenu cet article parce qu'il évoque le concept de représentation destinée aux algorithmes (équilibrage & association). En effet, le résultat de la fusion est utilisé par l'algorithme de classification pour détecter des objets. La plupart des articles en télédétection utilisent la fusion de données principalement pour la présentation des résultats à l'usager ou alors pour générer de nouvelles images, mais rarement pour utiliser le concept d'association.

De plus, ce type de fusion est particulier, car il aborde le problème des sources de données incompatibles (données matricielles vs données ponctuelles). En effet, pour pouvoir superposer l'image LIDAR sur l'image CIR correctement, il a fallu :

- Déterminer la position et l'orientation exactes des deux images par rapport à une référence universelle et fiable.
- Appliquer un algorithme de sélection et de fusion des pixels

La fusion de pixel ne change pas la nature même de l'information contenue dans le canal, mais peut induire une détérioration de la qualité du signal original. La méthode utilisée pour cette fusion n'est pas détaillée et laisse croire qu'il s'agit d'un placement du dernier candidat. Le candidat est le pixel de l'image LIDAR qui se voit assigner une place dans un pixel de l'image CIR. Le placement du dernier candidat signifie que c'est le dernier candidat à être placé qui retient la place dans l'image CIR. En d'autres mots, un candidat pour un pixel écrase celui qui y était assigné avant lui. C'est une méthode rapide et communément utilisée dans le domaine.

3.1.2 Fusion des analyses

La fusion des analyses consiste à utiliser les résultats de différentes analyses comme la classification, la régression ou la discrimination pour constituer une nouvelle analyse qui tienne compte de tous ces résultats, de manière à nous guider vers une meilleure décision.

Les travaux de Streilein sont un bon exemple de fusion des analyses. Son article intitulé « Fused Multi-Sensor Image Mining for Feature Foundation Data » [38] présente une méthode où la fusion de données est effectuée sur les analyses elles-mêmes. Pour différents capteurs, il extrait différentes caractéristiques qu'il compile ensuite au moyen d'un algorithme de fusion de données.

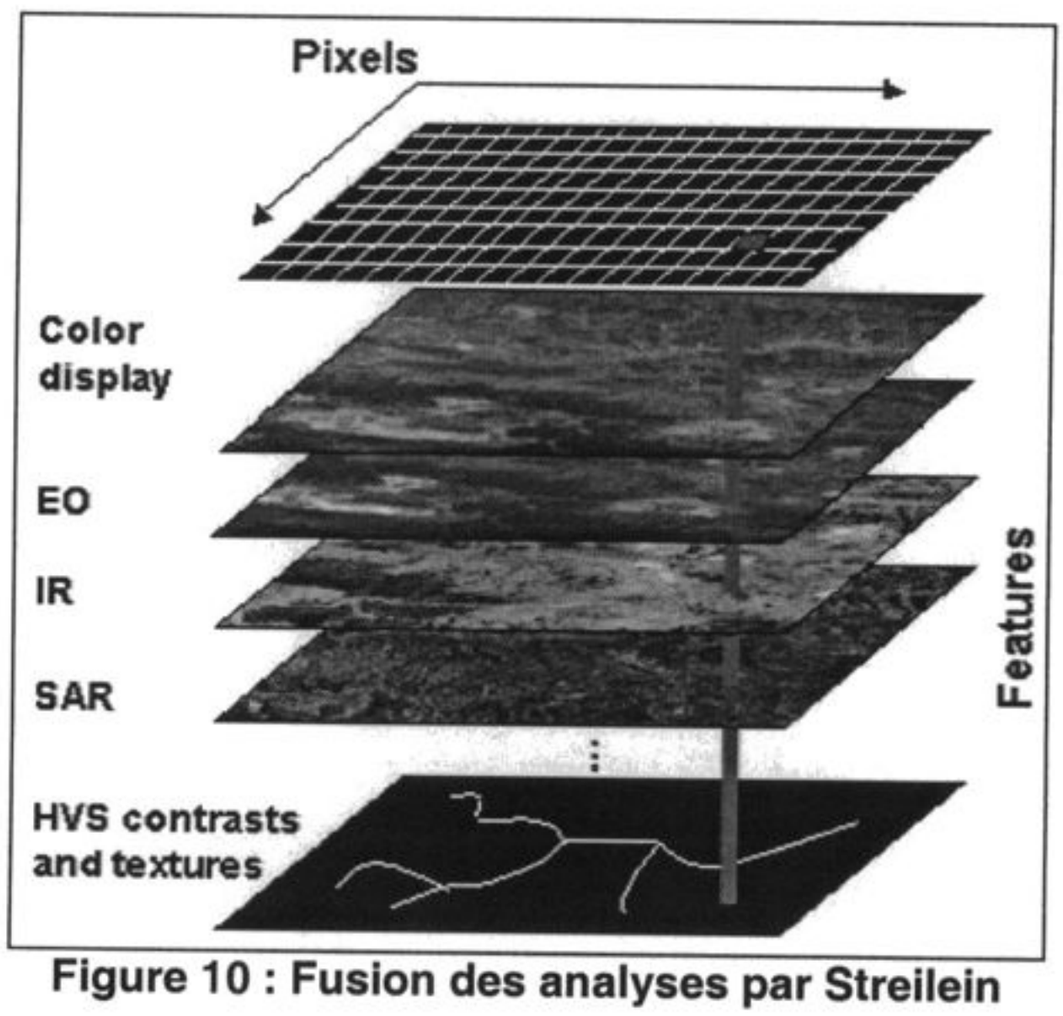


Figure 10 : Fusion des analyses par Streilein

Le résultat de la fusion permet alors à un algorithme subséquent d'effectuer une classification efficace des objets d'intérêt. La figure précédente (Figure) illustre bien le processus utilisé.

3.1.3 Fusion des représentations

La fusion des vues concerne à la fois l'usager et les algorithmes. En effet, si la vue résultante de cette fusion est une représentation destinée à l'usager, alors cette fusion améliore la perception des objets pour le système sensoriel humain. Par contre, si la fusion est destinée aux algorithmes, la vue fusionnée présente des caractéristiques intéressantes pour l'algorithme auquel elle est destinée.

Augmentation de la résolution par fusion de données

Un article écrit par M. Gonzalez [27] présente une méthode de fusion de données dont le but est d'améliorer la résolution d'images basse résolution à partir d'une seule image haute résolution. La technique consiste à ramener les bandes basse résolution à la même taille que la bande haute résolution, puis de combiner les décompositions en ondelettes des résolutions respectives pour ajouter les détails manquants provenant de l'image haute résolution dans les images basse résolution. La figure suivante tirée de l'article est fournie à titre informatif.

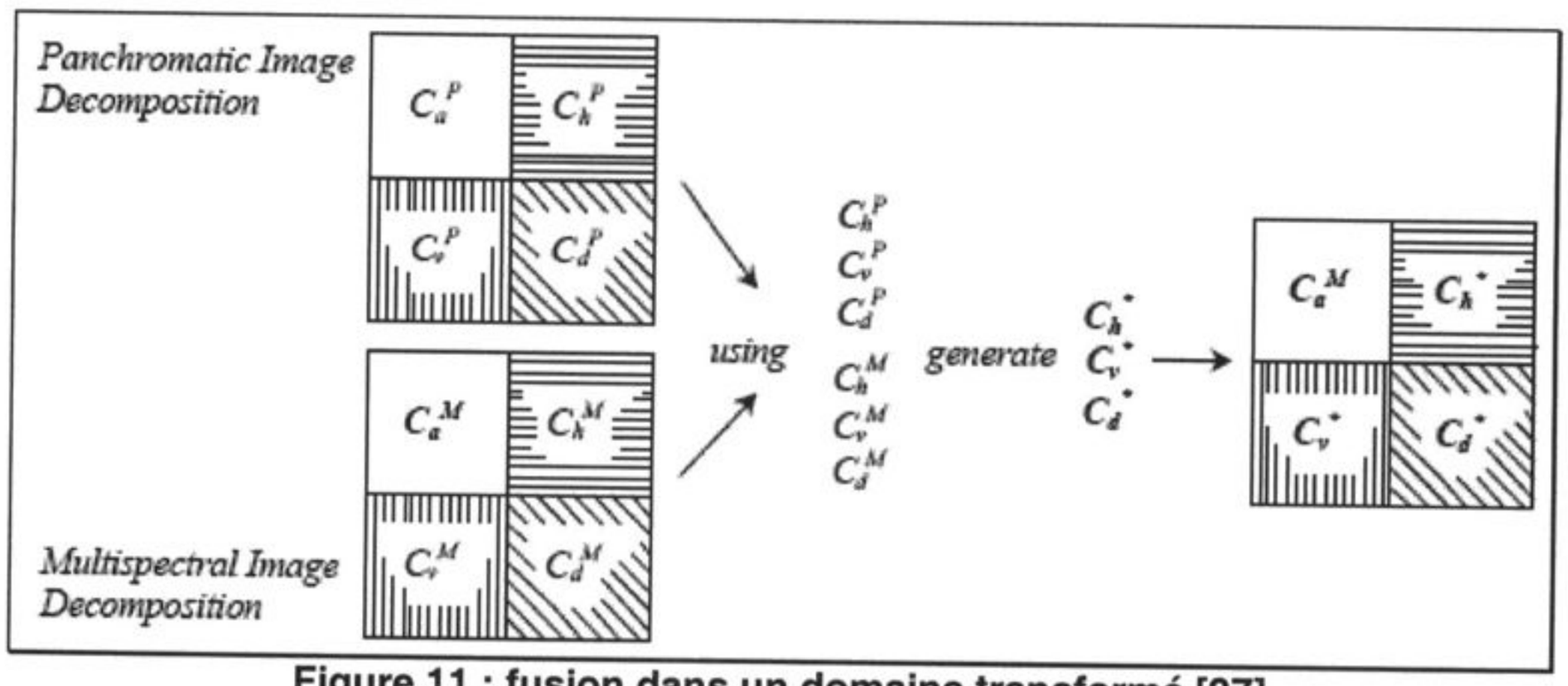


Figure 11 : fusion dans un domaine transformé [27]

La méthode propose d'utiliser la décomposition en ondelettes pour augmenter la résolution d'une image basse résolution. La méthode est conçue spécialement pour les images satellitaires parce qu'elle utilise le canal panchromatique pour améliorer la résolution des autres canaux à basse résolution. Il faut savoir que le canal panchromatique a une résolution double qualité par rapport aux autres. En travaillant sur les quatre spectres ondelettes décomposés, l'auteur de la méthode suggère que l'on puisse raffiner les détails des images basse résolution avec plus de précision. Meenakshisundaram [28] a d'ailleurs utilisé cette idée dans la réalisation de son mémoire de maîtrise pour détecter avec plus de précision des routes difficiles à repérer.

La technique proposée est une fusion de représentations, car elle utilise deux représentations différentes d'une même région pour en produire une nouvelle, améliorée. Comme les travaux ont été réalisés sur les images matricielles d'origine, avec seulement une simple conversion des images en matrices, l'utilisation d'un index spatial n'a pas été requis. Si, par contre, nous aimerions utiliser la même méthode de fusion de représentations sur d'autres types d'images (ponctuelles, vectorielles?) l'utilisation d'un index deviendrait alors obligatoire.

3.1.4 Une architecture de fusion de données

Publié en 1998, l'article de James Llinas intitulé "An Introduction to Multi-Sensor Data Fusion" [25] est un des seuls rencontrés à proposer une architecture relativement détaillée pour exécuter la fusion de données de manière plus générale. La figure suivante (Figure) est tirée de l'article et exprime bien le travail de modélisation du processus de fusion de données qui a été réalisé par l'auteur.

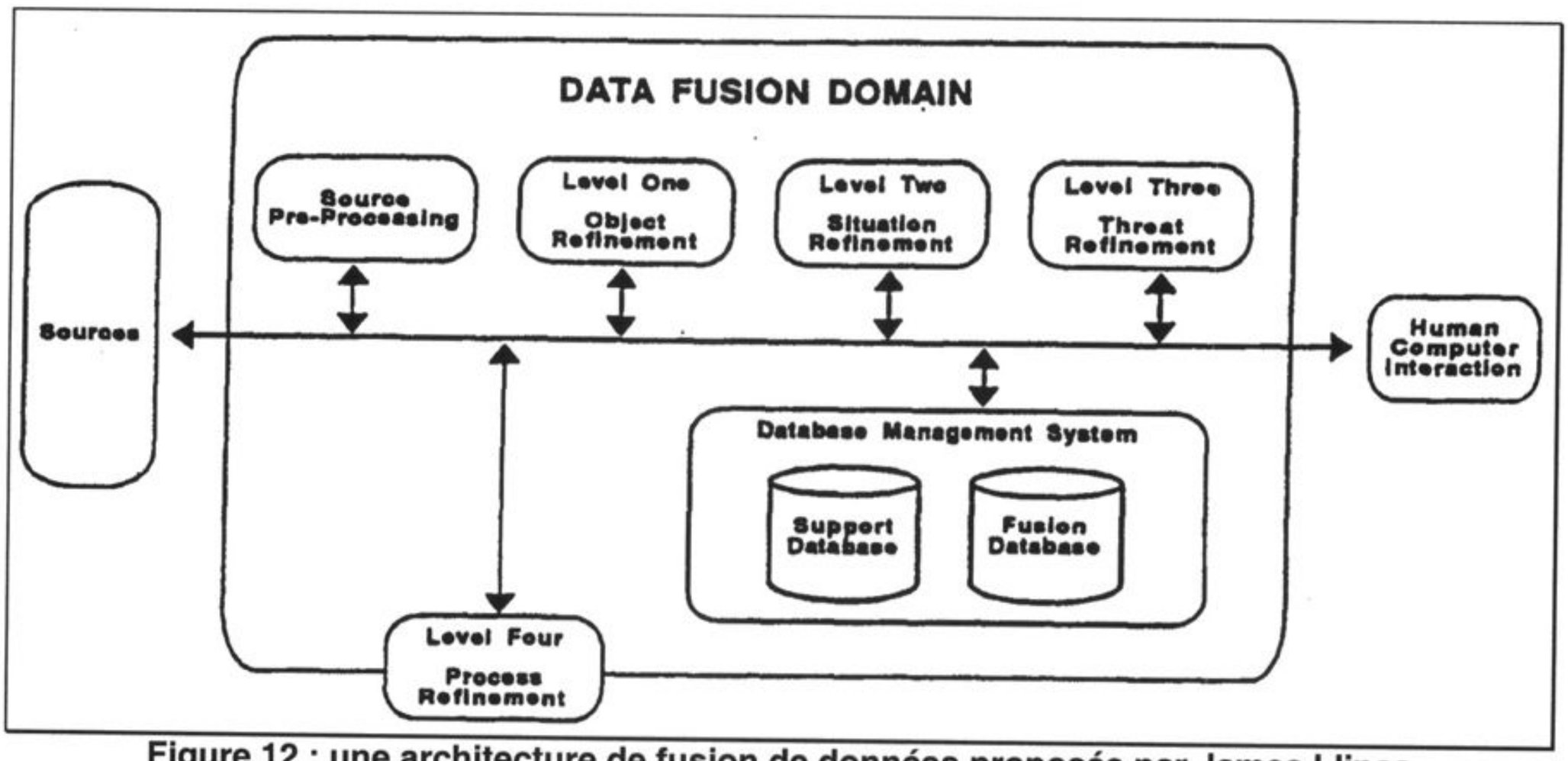


Figure 12 : une architecture de fusion de données proposée par James Llinas

Dans la structure proposée, l'auteur tient compte de la possibilité d'avoir plusieurs sources et de les modéliser comme tel, ce qui est un point absent de la plupart des articles traitant de la fusion de données. On retrouve aussi un autre aspect important : la présence d'un support de base de données intégré à la structure. Nous verrons dans la section méthodologie de ce mémoire que cet article est l'un des plus inspirants et des plus concrets pour résoudre le problème proposé en introduction.

3.1.5 Autres ouvrages connexes

Les ouvrages précédents ont été sélectionnés pour leurs similarités avec les différents projets sur lesquels nous travaillons déjà, mais ils ne sont pas les seuls ouvrages qui ont été consultés lors de ce recensement de la littérature en fusion de données.

Relativement au problème décrit en introduction, nous n'avons pas rencontré de projet dévoué spécialement à la réalisation d'une architecture de fusion de données dans le domaine de la télédétection. Dans tous les articles sur la télédétection qui ont été retenus, les travaux de fusion de données sont réalisés avec le but bien précis d'atteindre les objectifs du travail scientifique. Ainsi, on retrouve plusieurs éléments du domaine de la fusion de données éparpillés sur plusieurs projets scientifiques. De plus, la manière dont sont fusionnées les données est parfois marginalisée au point de n'en parler que dans une seule phrase!

Bien qu'aucun projet ou article n'ait proposé de solution intégrale au problème de la fusion de données dans le domaine de la télédétection, la communauté scientifique propose plusieurs méthodes et plusieurs idées pour y parvenir. Dans le cadre de ce mémoire, nous allons utiliser ces idées de manière à constituer un projet tangible et dédié à cette œuvre. Pour ces contributions, nous aimerions citer les travaux suivants que nous avons trouvés particulièrement pertinents :

Table 4 : quelques articles pertinents dignes de mention

Type de travaux	***Références***
Fusion d'attributs de différentes résolutions	[6] [8] [21] [32] [19]
Fusion d'attributs de même résolution	[10] [38] [40]
Fusions pyramidales d'un attribut	[34]
Fusion de capteurs	[37]
Fusion de données (général)	[39]
Fusion de données (architectures)	[15] [14]
Fusion de capteurs (architectures)	[23]

Il existe d'autres articles où les chercheurs s'intéressent davantage à la fusion elle-même qu'à son impact au sein de chaînes de traitement, mais ils sont du domaine des bases de données et se trouvaient parfois hors contexte.

3.2 Les index spatiaux et la génération de représentations

La génération de représentations, ou encore « vues », telles qu'on l'entend ici correspond à la production d'images matricielles présentant le contenu fusionné de plusieurs sources ou capteurs sous une seule structure de référence. Le but d'un tel travail est de permettre la création d'algorithmes génériques qui puissent fonctionner sur plusieurs sources différentes et avec d'autres algorithmes.

La génération de représentations est une fusion de données sur les attributs. C'est un sous-ensemble de la fusion d'attributs et on la retrouve principalement dans les logiciels de traitement d'images sous le nom de « layers » (couches) ou encore « blending » (mélanges).

Fondamentalement, la réalisation d'une représentation est fondée sur l'index spatial utilisé. On l'appelle aussi « spatial index » en référence aux index des bases de données. La signification de ce dernier est absolument la même que pour le premier et il est bienvenu dans le domaine de la télédétection. Il existe plusieurs techniques pour réaliser des index spatiaux. De manière générale, on retrouve trois catégories d'index :

- Tables de recherche et fonctions de hachage (lookup tables / hashing)
- Grilles (*Grid*)
- Arbres de recherche (arbres quaternaires, arbres R…)

Dans notre contexte, l'utilisation de ces techniques d'indexage est limitée à un objectif précis : présenter à l'utilisateur (humain ou algorithme) une représentation uniforme et universelle d'un ensemble d'images sources. Nous aimerions que l'utilisateur n'ait qu'à travailler avec une représentation matricielle simple qui rende les traitements complexes de fusion de représentations des sources transparente.

De plus, à la demande de l'utilisateur la représentation obtenue devrait permettre de faire abstraction de la position réelle et exacte de chaque donnée. Par exemple, l'utilisation de numéro de lignes et de colonnes plutôt que de coordonnées géospatiales. Pour que cette traduction entre les représentations réelles et la représentation universelle se fasse correctement, un index spatial est requis.

La principale raison pour laquelle ce genre d'indexage est requis est que l'accès aux données spatiales n'a rien de précis. Les algorithmes sont beaucoup plus efficaces sur des matrices bien alignées et de résolution moyenne que sur des points individuels désalignés localisés à une résolution maximale. Générer une vue permet de rendre le travail aux algorithmes beaucoup plus facile.

3.2.1 Tables de recherche et fonctions de hachage

Les tables de recherche (lookup tables) sont une matérialisation des fonctions de hachage. On choisit une fonction de hachage qui nous avantage et, moyennant que les données sources soient statiques, on construit une table de recherche. Cette table de recherche va nous permettre d'accélérer le fonctionnement du système dans lequel il est implanté.

En télédétection, les tables de recherche requièrent une fonction de hachage particulière qui puisse déterminer des régions exploitables dans l'image source. Un exemple fréquent de l'utilisation d'une table de recherche consiste à trier les données sources et à stocker dans une table une référence vers chaque groupe de données triées. Ainsi, lorsqu'on veut avoir accès à des données, on trouve l'item le plus près dans la table de recherche, qui nous indique la position du groupe de données dans lequel chercher. Cela a pour effet de localiser la recherche des données désirées et aboutit à de meilleures performances.

Une table de recherche qui comporte autant d'entrées qu'il y a de régions spatiales correspond conceptuellement à une grille.

Une table de recherche dont les références accèdent à d'autres tables de recherches correspond à un arbre de recherche.

Extendible hashing

Proposée par Ronald Fagin [9], cette méthode combine les performances d'une table de hachage avec l'arbre binaire non équilibré. Les auteurs partent du principe que si un arbre binaire n'est pas équilibré, il est possible de le rendre équilibré en mettant en place une table de hachage qui puisse donner accès directement aux noeuds internes de l'arbre. De cette manière, la table de hachage devient extensible par les propriétés de l'arbre binaire, et l'arbre binaire devient équilibré à cause de l'accès rapide aux noeuds internes.

3.2.2 Grilles

Les grilles sont la forme la plus simple d'une représentation : chaque cellule contient une référence directe aux données qu'elle contient. Elles sont particulièrement utiles lorsque les sources sont de types matriciels. Par contre, lorsque les sources ne sont pas matricielles, les grilles induisent des pertes considérables en espace mémoire dues au fait que les cellules ne sont pas toutes remplies. On retrouve d'ailleurs le même genre de problème dans certains arbres.

Les grilles sont souvent réservées à l'affichage dans les logiciels de traitement d'images, car elles constituent la dernière étape entre la préparation et l'affichage. Les images affichées à l'écran sont toujours matricielles, il y a donc toujours besoin de convertir ce qui ne cadre pas.

Grille à baquets

Les grilles à baquets (b*ucket grid*) sont des grilles dont chaque cellule pourrait être vue comme un panier. Contrairement aux grilles de base, chaque cellule peut contenir plus d'une donnée. Ce type de grille est utilisé spécialement en segmentation, où on réduit volontairement la résolution d'une image pour travailler sur « l'ensemble » et ensuite retravailler chaque cellule et son contenu indépendamment.

Grille

La grille [29] (*grid file*) est une technique qui consiste à partitionner un fichier de données selon un attribut de sorte à pouvoir accéder rapidement aux données qui possèdent des attributs semblables. Appliquer ce concept à la télédétection est équivalent à trier un fichier par les dimensions disponibles. Le principal avantage de cette méthode est de garantir l'accès aux données en seulement deux accès au disque. L'inconvénient de cette méthode est qu'il arrive parfois qu'une quantité exorbitante de cellules se retrouvent vides, gaspillant les ressources.

La **grille double** [18] (twin grid file) est une modification apportée à la grille où l'espace utilisé est optimisé en permettant à des cellules d'être subdivisées si elles sont trop pleines.

Il existe aussi la grille à base du **PLOP Hashing** [22] qui élimine la nécessité de maintenir un dictionnaire d'accès en le remplaçant par une fonction de hachage.

Grilles basées sur l'interpolation

Les grilles interpolées [13, 31] (*interpolation-based grid files*) sont des modifications apportées à la grille sur la manière dont sont créées les partitions. L'objectif principal est de créer le moins de partitions vides possible.

Le **fichier BANG** [13] ajoute la possibilité de créer des partitions qui peuvent contenir d'autres partitions.

3.2.3 Arbres de recherche

Les arbres de recherche constituent probablement le domaine de recherche le plus effervescent des trois, stimulé par la croissance rapide du domaine des bases de données. En effet, les bases de données sont toujours à l'affût de meilleurs moyens d'accéder aux données !

Arbre quaternaire

Le concept d'arbre quaternaire a été utilisé et décrit en 1974 par Finkel et Bentley [11] comme étant un arbre à quatre branches où chaque nœud représente un élément spatial à représenter. Ainsi, à chaque nœud, on peut indiquer la position spatiale de quatre autres nœuds en correspondance avec les quatre quadrants créés : nord-est, sud-est, sud-ouest et nord-ouest. En partant de la racine de l'arbre, on peut naviguer dans un mouvement s'apparentant à une spirale en allant de nœud en nœud jusqu'à ce qu'on ait trouvé l'information qui nous intéressait. À cause que chaque nœud subdivise l'espace en quatre quadrants, la spirale se réduit rapidement à un emplacement précis.

L'arbre quaternaire proposé par les auteurs est inspiré de l'arbre binaire équilibré (*binary balanced tree)*. Chaque fois qu'il y subdivision de l'arbre, un algorithme maintient l'arbre dans un état équilibré. Cela garantit que le temps d'accès à toute donnée répertoriée dans cet arbre soit le même.

Comme nous le verrons plus loin, la communauté scientifique s'est inspirée de ce concept d'arbre quaternaire pour créer trois cas d'utilisations typiques : les points, les régions et les lignes (vecteurs).

Arbre quaternaire à point

Techniquement, l'arbre quaternaire à point (*point quadtree*) fonctionne de la même manière que l'arbre quaternaire décrit précédemment. De ce point de vue, nous pourrions dire qu'il s'agit de synonymes. La raison pour laquelle cette catégorie a été créée est pour la différencier d'autres méthodes qui sont apparues par la suite.

Les arbres quaternaires à point désignent toutes les méthodes qui se servent des nœuds de l'arbre de recherche pour stocker les éléments atomiques qui font l'objet de la recherche. Plus simplement : il s'agit de points. Un point est un objet composé d'une position (X,Y) et d'une valeur (intensité). Il est possible de rencontrer d'autres sortes de « points » mais le concept de l'arbre quaternaire à point demeure le même.

De même, lorsqu'on parle d'un arbre quaternaire à point en général, on désigne l'œuvre de Finkel et Bentley.

Arbre quaternaire à région

Les arbres quaternaires à région (r*egion quadtree*) diffèrent de l'arbre quaternaire à point de la manière dont les nœuds de l'arbre sont utilisés. Cette fois, la donnée est stockée dans les feuilles de l'arbre et les nœuds servent uniquement à partitionner et à subdiviser l'espace en régions. La manière dont l'arbre est parcouru est la même, mais les données sont localisées directement, sans être comparées à d'autres. Les arbres quaternaires à région sont pour cette raison moins dépendants d'algorithmes d'équilibrage parce que la structure générale de l'arbre demeure constante. Les régions sont toujours les mêmes, ce ne sont que les feuilles qui changent.

L'**arbre quaternaire MX** ou **arbre quaternaire matriciel** (*MX quadtree*)subdivise chaque région strictement en quatre sous régions de taille égale.

Il est plus performant sur des données où la taille spatiale est définie et où la résolution fait en sorte que l'arbre puisse être stocké en mémoire. Ce type d'arbre prend beaucoup de mémoire à cause qu'il doit stocker le réseau de noeuds qui détermine les régions en plus des données elles-mêmes. Les arbres quaternaires à point ont l'avantage de stocker les données dans les noeuds, ce qui économise de l'espace. Ce genre d'arbre est aussi appelé *Trie*, parce que le réseau de noeuds ne contient pas les données. Pour un arbre quaternaire matriciel, le nombre de régions est déterminé par le niveau de précision désiré. Ainsi pour un arbre de niveau 2, nous aurons un total de 16 régions, qu'elles soient vides ou non. Il existe une variante où les branches de l'arbre ne sont construites que si elles mènent à une donnée, économisant ainsi beaucoup de mémoire.

L'**arbre quaternaire PR** [35] ou **arbre quaternaire point-région** (*PR quadtree*) se distingue de l'arbre quaternaire matriciel en ce sens que bien que les points soient aussi stockés dans les feuilles de l'arbre, la subdivision des régions n'est pas uniforme : l'algorithme tentera de faire en sorte que chaque point soit stocké dans la plus grande région possible, évitant ainsi de créer des noeuds pour toutes les régions possibles comme c'est le cas pour l'arbre quaternaire matriciel. Nous obtenons alors des données stockées à des niveaux différents de l'arbre. Ce type d'arbre est beaucoup plus économique en mémoire que l'arbre quaternaire matriciel.

Arbre quaternaire à arêtes

Les arbres quaternaires à arêtes (*edge quadtrees*) sont utilisés exclusivement pour la représentation spatiale de segments de droite et de vecteurs. Cette catégorie d'arbres quaternaires ne sera pas discutée dans le cadre de ce travail, car elle ne concerne pas le problème à l'étude. Elle est discutée et bien illustrée par Lindenbaum et Samet [24].

Arbre B

L'arbre B [2] est un arbre équilibré où toutes les feuilles ont la même hauteur. La principale caractéristique de cet arbre est que tous les noeuds sont pleins à au moins 50%. Les arbres B sont construits à partir d'un nombre premier k qui définit les nombres minimal (k+1) et maximal (2k+1) d'enfants par noeud.

L'arbre B+ est une modification de l'arbre B qui ajoute un pointeur à chaque feuille de l'arbre pour permettre le passage rapide entre les feuilles. L'arbre B nécessite normalement de passer par le parent d'une feuille pour rejoindre la prochaine.

Arbre K-d

L'arbre k-d [3, 4] est une modification de l'arbre BSP (Binary Space Partitioning). Il s'agit d'un arbre multidimensionnel, d'où son nom k-d où k désigne l'ordre de la dimension. Son principe de fonctionnement est que les données sont stockées dans les noeuds et la division de l'espace se fait en créant des plans qui regroupent les données selon leur adjacence par rapport aux autres. Contrairement à l'arbre BSP, les plans doivent être orthogonaux par rapport aux axes du système de coordonnées utilisé.

L'arbre K-D-B [33] tire partie des avantages de l'arbre B et de l'arbre k-d pour en former un nouveau.

L'arbre hB [26] est une modification de l'arbre K-D-B. Les principales différences sont : les noeuds sont organisés comme des arbres k-d (oublions l'arbre B) et la méthode de décision menant à la division d'un noeud (*node splitting*) a été étendue pour pouvoir tirer partie de plus d'un attribut.

Arbre R

L'arbre R [16] (*R-Tree*) a été spécialement conçu pour les données spatiales. C'est un arbre équilibré où chaque niveau nous donne de l'information sur la région occupée par les niveaux enfants. Ainsi, pour effectuer une recherche spatiale, il suffit de descendre dans l'arbre chaque fois que la requête répond au critère d'inclusion de la région (branche) examinée. Les points se trouvent aux feuilles et il est possible de voir des régions se superposer (donc plus d'une branche à parcourir pour une même requête).

Le **Buddy Tree** [36] reprend quelques concepts de partitionnement de la grille pour le rendre plus performant. Entre autres, l'arbre produit des régions qui ne se superposent pas (contrairement à l'arbre R) et ne stocke pas de partitions vides (contrairement à la grille). Il en résulte un arbre de recherche assez performant.

3.2.4 Note sur les représentations vectorielles

En dessin assisté par ordinateur [12], les représentations vectorielles sont très importantes. Leur structure de base est constituée de points auxquels on annexe une topologie. La topologie peut varier entre trois formes : arêtes, surfaces ou objets. L'utilisateur peut alors utiliser un logiciel de dessin spécialisé pour réaliser des formes complexes.

La représentation vectorielle est une représentation qui est valide seulement en concept. Pour l'afficher à l'écran ou pour la traiter par un algorithme, l'image vectorielle doit être soumise à un processus de rastérisation qui consiste à convertir vecteurs en pixels.

Notre travail ne portera pas sur les outils vectoriels ou les représentations vectorielles et cette note avait simplement pour but de clarifier les choses à ce sujet.

4 Méthodologie

La plateforme Feature Ex-RA a été développée pour réaliser nos tests sur les index spatiaux. Elle intègre le moteur de fusion de données qui lui-même intègre les différents index spatiaux. Vous trouverez des détails quant à l'architecture de cette plateforme à la section 4.1 (La plateforme Feature Ex-RA) de ce document.

Pour déterminer quels devaient être les différents critères d'évaluation des index, nous nous sommes penchés sur ce qu'étaient les principales qualités recherchées lorsqu'on fait du prototypage en télédétection. Nous en sommes venus à la conclusion que ces qualités sont :

- un chargement et un déchargement rapide du moteur
- un temps d'exécution des requêtes court
- une facilité à modifier les algorithmes pour les tester.
- une utilisation limitée de la mémoire

Ces qualités ont été traduites en critères plus formels à la section 4.4 ci-dessous (Mesures de performance).

4.1 La plateforme Feature Ex-RA

Afin de pallier aux lacunes que présentent les plateformes commerciales, nous avons étudié l'architecture de Llinas et proposé une adaptation concrète et générale au domaine de la télédétection. L'architecture respecte les principes du génie logiciel et propose non seulement le design d'un noyau de fusion de représentations, mais aussi apporte une spécification globale de l'interfaçage que cette architecture a avec les différentes sources de données et avec les algorithmes.

Le noyau de fusion a été conçu pour permettre le remplacement facile de la technique d'indexage spatial à l'essai. La figure ci-dessous illustre rapidement la facilité avec laquelle il est possible d'interchanger l'index spatial à volonté, sous la forme de plugins :

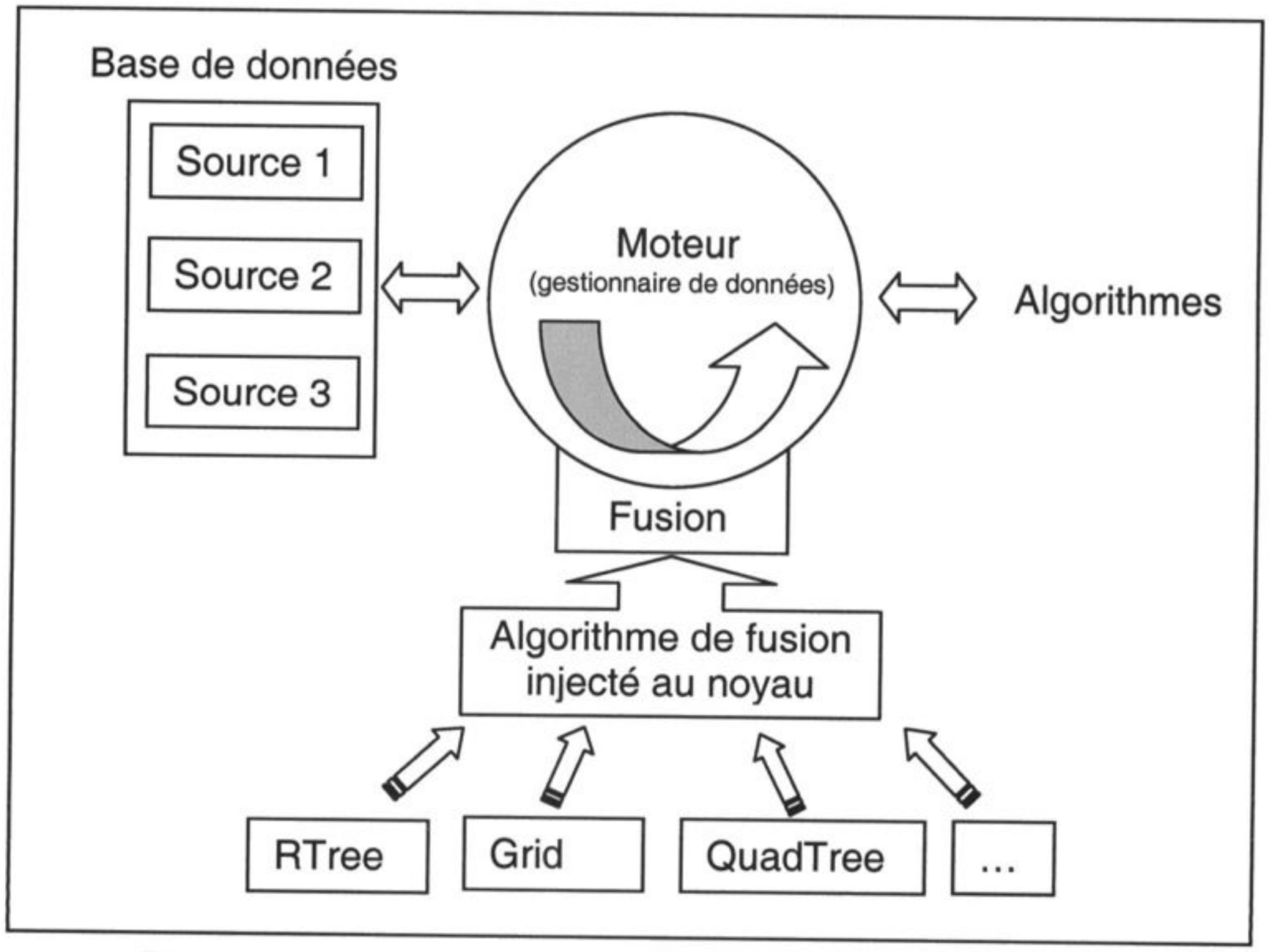


Figure 13 : architecture du noyau (modèle à injection de code)

Les interfaces d'entrée-sortie au moteur de fusion ont été modélisées sous trois formes principales : les sources de données, les algorithmes et les vues-utilisateur. L'architecture est suffisamment générale pour permettre l'implémentation rapide des algorithmes.

Chacun des index spatiaux choisis a été implémenté et inséré à l'architecture. Au niveau des tests de performance et d'efficacité, les différentes méthodes ont été soumises à des bancs d'essai identiques et documentés. Plus de détails peuvent être trouvés à la section résultats de ce document.

4.1.1 Architecture générale

L'architecture de Llinas proposait quatre entités de base : les sources, la base de données interne, le moteur de fusion de représentations et les vues utilisateur. À défaut d'avoir le code source de l'architecture de Llinas, le même principe a été implanté dans la plateforme Feature Ex-RA :

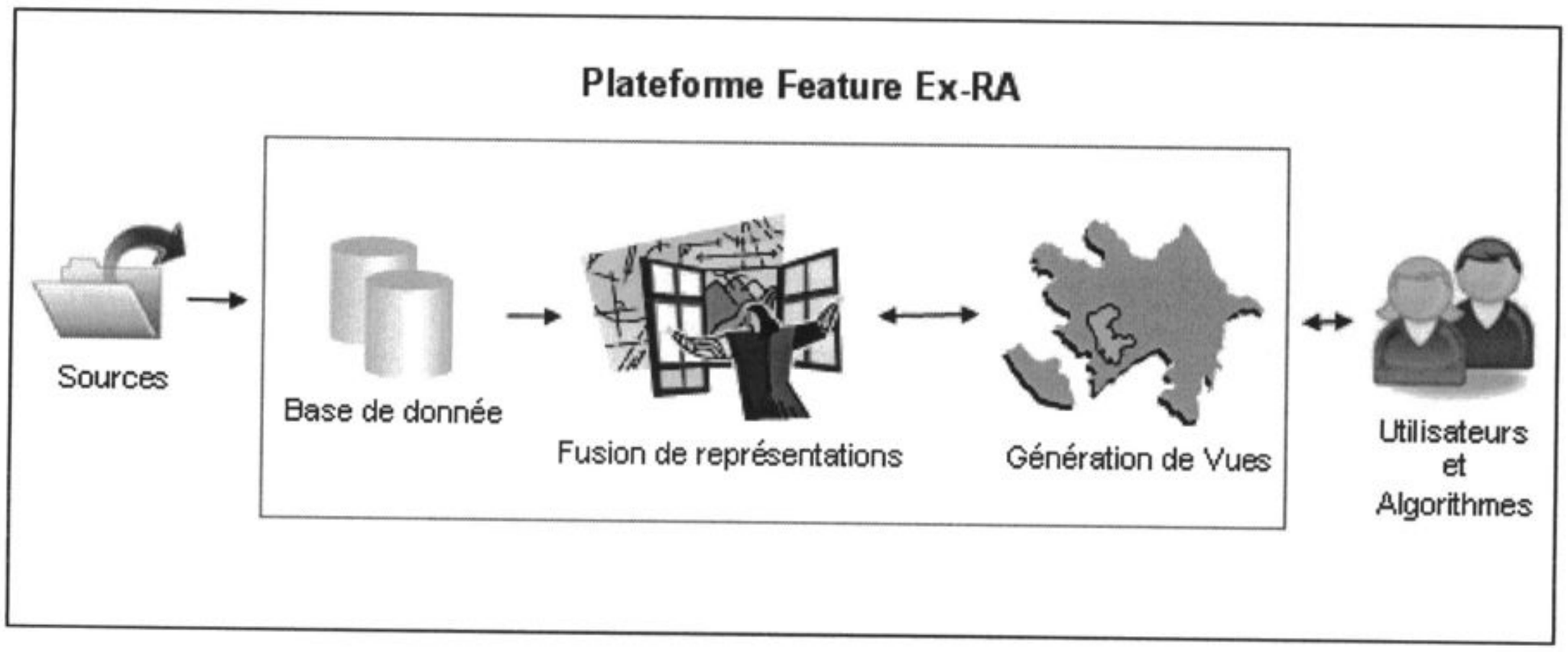


Figure 4-B : architecture générale

La figure précédente montre bien le processus par lequel les données prennent forme avant d'être utilisées. Tout d'abord, la base de données transforme les sources de données en une information uniformément accessible par le système. Ensuite, la fusion de représentations interprète cette information et lui donne un sens. Finalement, une vue est générée à la demande de l'utilisateur ou de l'algorithme, vue qui fait abstraction des complexités retrouvées dans les premières étapes.

4.1.2 Source de données

La plateforme est conçue pour pouvoir lire différentes sources de données sans leur imposer d'altération ou de conversions. En fonctionnant sur une base de points, la base de données peut stocker n'importe quel type de source (ponctuel, matriciel ou vectoriel) sans altérations ou pertes d'information.

En effet, une source matricielle ou une source vectorielle peuvent être représentées comme un nuage de points. La première n'est qu'une série de points alignés dans l'espace. La résolution de celle-ci peut être déterminée par l'espacement entre les points. La deuxième peut être représentée par une série de points et par une information complémentaire stockée sous forme d'un attribut.

Pour bien représenter les sources de données, outre la représentation ponctuelle, la plateforme utilise une structure de données multi-dimensionnelle généraliste. Il existe deux entités principales : les blocs de données et les blocs de position. Les blocs de données contiennent les attributs des données, alors que les blocs de position stockent l'information spatiale relative aux données. La séparation entre la position et la donnée est importante, car elle permet des manipulations

géométriques très rapides. Les données sont alors statiques et les blocs de position peuvent être réutilisés pour plusieurs blocs de données.

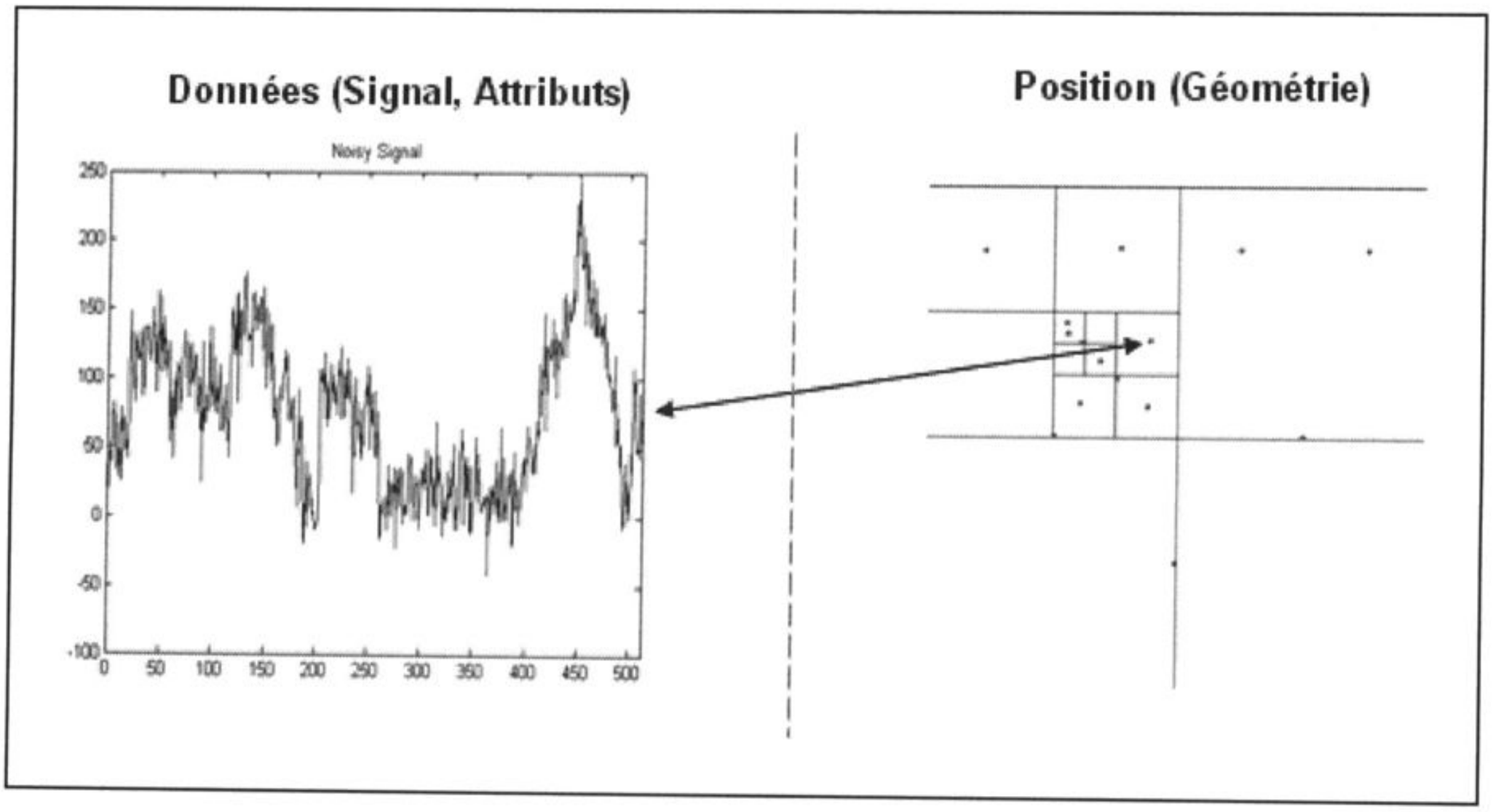


Figure 4-P : Relation entre les données et la position

Sur la figure précédente, on retrouve dans un bloc de données l'intensité d'un signal par rapport au temps. En soi, le signal ne nous dit pas grand-chose. Il faut le lier à sa position géographique pour lui donner un sens. Par exemple, s'il s'agissait d'enregistrements séismologiques, ils n'auraient de sens que si on les associe à leur position géographique.

Les blocs de positionnement peuvent comporter autant de dimensions qu'il le faut, permettant alors de stocker de l'information non seulement spatiale, mais aussi temporelle ou autre.

Les blocs de données peuvent comporter autant d'attributs que la source en fournit. N'importe quel attribut peut être déplacé dans un bloc de position

correspondant, dépendamment du sens qu'on lui donne. Par exemple, si la source fournit un identificateur spécial pour séparer les données en sections, il peut être utilisé comme dimension additionnelle, permettant alors de répertorier les données correctement.

En plus de cette structure de données flexible, la plateforme utilise un système de plugins génériques. On dit que les plugins sont génériques parce qu'ils possèdent la même structure que les plugins algorithmiques que nous verrons plus loin. Ainsi, chaque source de données peut être lue au moyen d'un plugin réalisé à cette fin. Par la suite, il suffit de mettre le plugin, compilé sous forme d'une bibliothèque dynamique, dans le répertoire prévu à cet effet. La plateforme reconnaît alors dynamiquement la présence du plugin, qui devient alors disponible à l'utilisateur.

Il est aussi très important de ne pas confondre ces plugins de lecture des sources avec les procédés d'importation. Il n'y a pas d'importation : les données sont lues dans leur format original et chargées dans la base de données comme une information brute, sans pertes de données. La distinction est importante, car la plupart des logiciels commerciaux imposent parfois l'importation des données, qui implique des pertes car il y a conversion. La conversion de données implique presque toujours une perte d'information.

4.1.3 Base de données

Feature Ex-RA est très souple quant à la base de données à utiliser : cela pourrait être une base de données réputée comme SQL Server et MySQL, ou alors cela peut être une base de données maison ou encore carrément stocké en mémoire vive.

Nous avons opté pour un stockage en mémoire des sources de données pour maximiser la performance des traitements. Les données à traiter sont choisies pour être raisonnables par rapport à la mémoire disponible.

4.1.4 Index spatiaux et fusion

Après le chargement des données vient le moment de les interpréter. Nous l'avons vu plus tôt, il doit y avoir d'abord le travail de l'indexage spatial, et ensuite une fusion de représentations. Une fois l'indexage spatial réalisé, la fusion de représentations devient triviale. La plateforme Feature Ex-RA utilise notamment la forme la plus simple de fusion de représentations : la fusion par ajout d'attributs. C'est-à-dire que tous les attributs rencontrés en entrée se retrouvent en sortie. À titre comparatif, d'autres types de fusion utilisent des attributs en entrée pour créer un ou plusieurs attributs en sortie. Dans ce cas, les attributs créés sont le résultat d'une fusion plus ou moins complexe des attributs en entrée. À l'intérieur de Feature Ex-RA, la complexité de l'algorithme de fusion a été tenue au minimum : les attributs ne sont pas altérés par la fusion. Cela nous permettra d'étudier l'impact seul du choix de l'index spatial sur les performances de la plateforme.

La plateforme est conçue pour permettre le remplacement rapide et simple de l'index spatial. Une fois l'index spatial implémenté, il devient automatiquement reconnu par la plateforme, laissant le choix à l'utilisateur de son utilisation au sein du moteur de fusion.

Par souci de réutilisation, certaines implémentations ont été adaptées à partir de code source existant. C'est le cas de l'arbre R [16, 17] et de l'arbre k-d [3, 43].

Au cours du développement, un index spatial a été créé. Il s'agit de la méthode des tables de recherches à enchaînements (*Chained Link Lookup Tables*) que nous désignons par la suite par tables d'enchaînements. Cette dernière méthode mérite quelques explications, car elle ne figure pas dans l'état de l'art et constitue une proposition nouvelle.

4.1.5 La méthode des tables d'enchaînements

La méthode des tables d'enchaînements est dérivée de la méthode de la grille à baquets : nous avons remarqué que cette méthode était rendue inefficace de par le nombre de petits objets à créer. En effet, une façon de réaliser une grille à baquets est de construire une grille de pointeurs, et de construire des petits vecteurs d'index pour les cellules concernées. La méthode est efficace pour les données ponctuelles, mais perd l'avantage dans le monde matriciel vis-à-vis la grille simple. Cette perte est due en grande partie au fait que la création des petits vecteurs d'index est extrêmement coûteuse. Ainsi, nous nous sommes penchés sur le problème et en sommes venus à la conclusion qu'il faudrait trouver un moyen d'allouer toute la mémoire en un seul coup, ce qui nous éviterait d'avoir à faire la maintenance sur les cellules individuelles. C'est exactement l'objectif de cette méthode : allouer toute la mémoire d'un coup, sans perdre la trace du contenu.

La méthode comporte trois tableaux de données : la table maîtresse, la table d'enchaînements et la table des index.

La table maîtresse correspond à la grille de pointeurs : elle ne contient qu'un élément par cellule et détermine une cellule de la table d'enchaînements. Si la cellule requise est vide, la table maîtresse contient un code indicateur à cet effet. Ainsi, l'accès à une cellule vide est toujours constant.

Ayant en main le numéro de cellule de la table d'enchaînements, on la consulte et on obtient deux valeurs : l'offset et le compte. L'offset est la position exacte des données qui nous intéressent dans la table d'index, et le compte indique combien de valeurs s'y trouvent stockées à la queue leu leu.

Finalement, nous accédons aux index en lisant directement à la cellule concernée. Si le compte indique plus d'un index, il suffit de lire les prochaines cellules de la table. L'accès aux données se fait donc en un temps constant de trois lectures. Notons que l'accès aux deuxièmes ou troisièmes données d'une série se fait en une seule passe et que c'est l'accès à une cellule qui coûte trois lectures.

Une lecture pour savoir que la cellule est vide, et trois lectures pour connaître le contenu. C'est une performance théorique qui se trouve dans les normes des autres index spatiaux. La vraie différence se fait sentir au niveau du temps d'allocation de la mémoire et au temps de réallocation. Le fait d'allouer un gros tableau d'un seul coup fait toute la différence face à la méthode de la grille à baquets, et ce, autant en mémoire que sur disque.

Cette méthode conserve les mêmes performances en écriture, à condition de connaître l'étendue exacte de l'information à stocker. Cette méthode n'est pas recommandée pour des situations où il serait préférable que la taille du stockage croisse progressivement.

4.1.6 Les vues et les snapshots

La génération des vues est la dernière étape de la manipulation des données avant qu'elles soient livrées aux algorithmes ou aux utilisateurs. Le terme vue est utilisé ici par allusion aux bases de données. Dans les bases de données, il est possible de créer des vues pour permettre aux utilisateurs de consulter les données sous une certaine forme standardisée. Le principe est le même pour la plateforme Feature Ex-RA : l'utilisateur travaille toujours sur une vue des données, jamais sur la base de données ni même l'index spatial.

Bref, l'utilisateur obtient l'accès à ses données avec une simplicité qui rappelle le logiciel Matlab : une vue est aussi simple à manipuler qu'une matrice. On obtient le nombre de rangées et de colonnes par le biais des fonctions width() et height() et on lit et écrit par le biais de fonctions get et set. Il y a aussi abstraction de la position géographique exacte : les lignes et les colonnes sont numérotées à partir de 0 avec des nombres entiers. L'accès à des positions géographiques exactes est toujours possible, mais ce n'est pas là l'intérêt qu'apporte la vue. De plus, la plupart des algorithmes fonctionnent sur des matrices à index entiers. La combinaison des numéros de ligne et de colonne forme ce qu'on appelle la coordonnée virtuelle et désigne une cellule spatiale dont les dimensions réelles sont gérées par la plateforme de fusion de représentations.

Par défaut, la génération de la vue se fait en temps réel : à chaque requête de lecture, la coordonnée virtuelle est propagée au travers du reste de la plateforme pour être traduite en données réelles. Si notre application fonctionne sur un bloc complet de données à la fois, il est possible alors d'utiliser les snapshots.

Les snapshots sont des prises de vues à un instant donné qui sont figées et stockées séparément. L'accès aux données devient alors instantané, avec un temps d'accès égal à une seule lecture. Pour le moment, les snapshots ne supportent pas le même concept en écriture, mais il est proposé de réaliser une sorte de mémoire tampon qui attendrait un ordre de confirmation avant d'écrire tout d'un seul bloc. Les snapshots sont très utile pour l'affichage des données sous forme d'image. Il est aussi possible de mettre à jour un snapshot sans pour autant le régénérer au complet.

4.1.7 Les algorithmes et les plugins

Nous l'avons vu un peu plus haut, la plateforme Feature Ex-RA utilise des plugins génériques pour permettre la lecture des sources de données. C'est valide aussi pour les algorithmes.

Les algorithmes peuvent être implémentés à la guise du chercheur, mais ils doivent se plier à l'interface générique. Cette interface prévoit une structure pour :

- Connaître le nom de l'algorithme et de l'information sur celui-ci.
- Connaître les paramètres requis et leur type.
- Transmettre les paramètres requis.
- Donner accès à certaines ressources propres à la plateforme (API)

Tout d'abord, en connaissant le nom et la catégorisation de l'algorithme, la plateforme est capable de classer et de présenter celui-ci à l'utilisateur.

Ensuite, en utilisant un mécanisme pour connaître les paramètres requis et leur signification, la plateforme peut solliciter les renseignements manquants à l'utilisateur par le biais d'un dialogue dynamique.

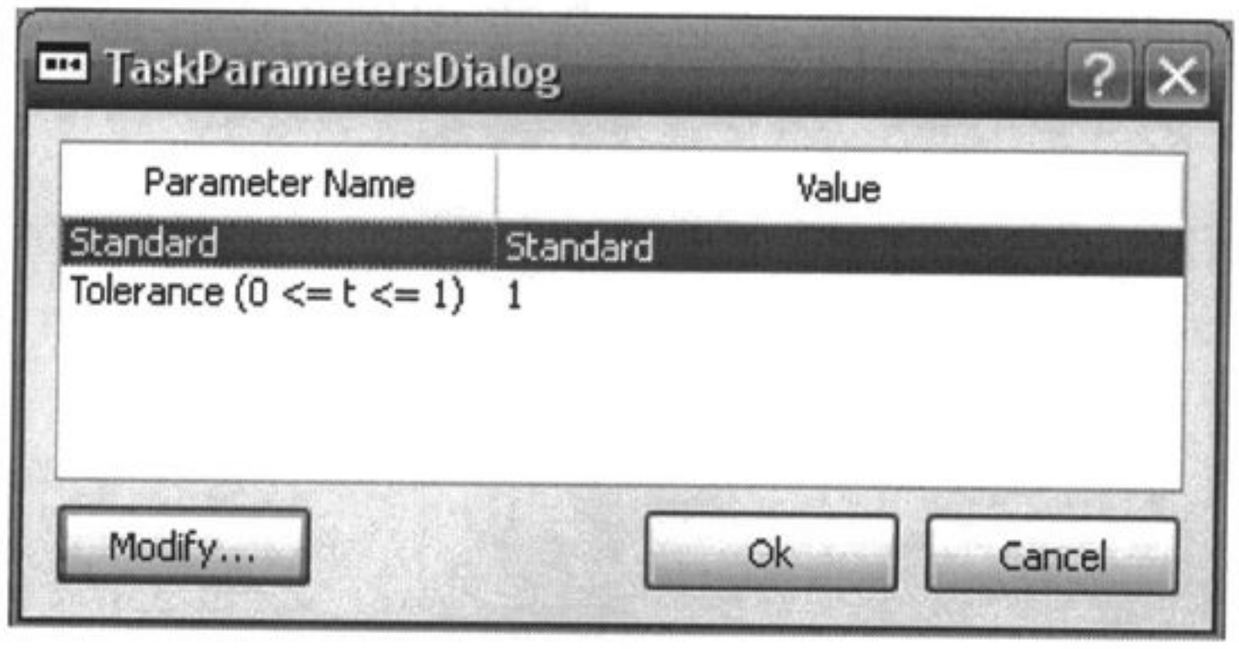


Figure 17 : Boîte de dialogue dynamique

Enfin, le mécanisme de lancement permet la configuration correcte de l'algorithme et donne accès aux ressources de la plateforme comme les échantillons, les vues, les snapshots et autres fonctionnalités.

Ainsi, en se conformant à ces règles de base, le chercheur encapsule ses algorithmes dans un cadre générique qui permet une réutilisation et un enchaînement faciles.

4.2 Les index spatiaux

Cette section comprend quelques informations importantes concernant l'implémentation des différents index spatiaux mis à l'épreuve au cours des tests. Tous les index spatiaux acceptent des requêtes de lecture et d'écriture, et travaillent sur les index de la base de données uniquement. Les index de la base de données sont traduits en information par d'autres mécanismes.

Arbre R

Implémentation réalisée par Antonin Guttman et adaptée progressivement par Michael Stonebraker, Melinda Green, Paul Brook et Greg Douglas [16, 17].

Arbre quaternaire matriciel

L'arbre quaternaire matriciel a été implémenté sous la forme d'un arbre à quatre branches. Seules les branches menant à une feuille sont créées. Les index sont placés aux feuilles. Un algorithme non récursif est utilisé pour déterminer à quelle branche doit être fixé un index, et son temps d'exécution dépend de la résolution demandée.

Grille

La plus simple des structures à implémenter, la grille a été représentée sous la forme d'une matrice d'entiers. Chaque entier représente un index de la base de données. La principale faiblesse de la grille est qu'il est impossible de stocker plus d'un index par cellule, résultat : pour obtenir une bonne qualité il faut s'assurer que la grille soit suffisamment fine pour ne stocker au plus qu'un seul élément par cellule. Beaucoup d'espaces vides sont créés durant ce processus (voir exemple ci-dessous).

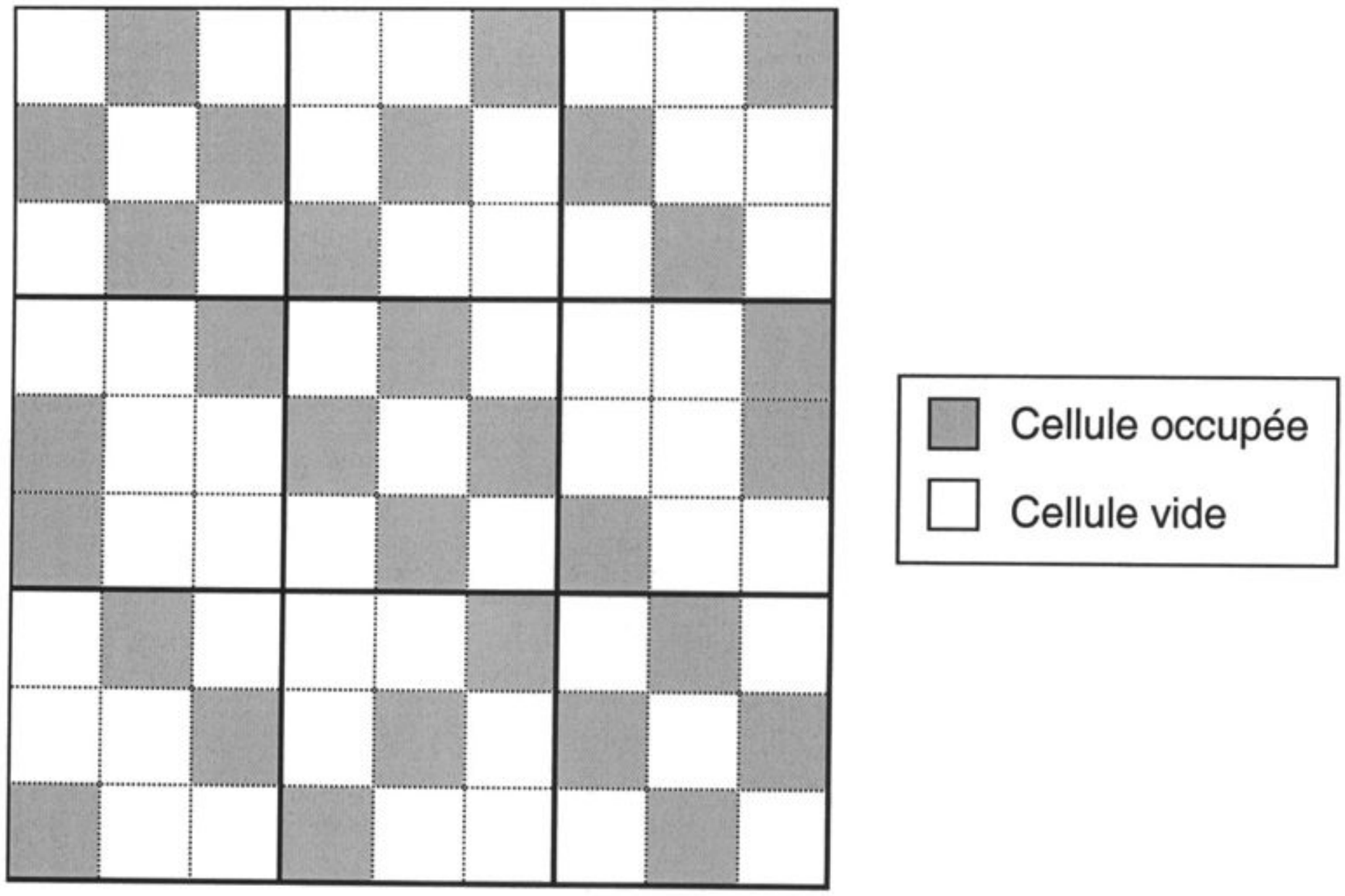


Figure 18 : L'utilisation d'une grille impliquant beaucoup de pertes

Grille à baquets

La grille à baquets vient pallier à la principale faiblesse de la grille simple en permettant le stockage de plusieurs index par cellule, limitant ainsi le nombre de cellules vides. Son implémentation a été réalisée sous la forme d'une matrice de pointeurs, chaque pointeur étant soit mis à NULL pour une cellule vide ou pointe vers un vecteur dans les autres cas.

Arbre k-d

L'arbre k-d a été implémenté exactement comme dans l'article scientifique, soit un arbre binaire avec plusieurs niveaux représentant les différentes dimensions. Les index sont stockés à même les nœuds.

Tables d'enchaînements

La méthode par la table d'enchaînements a été expliquée un peu plus haut. Elle est constituée de trois tables, dont l'une comprend tous les index à stocker. La figure suivante illustre son fonctionnement :

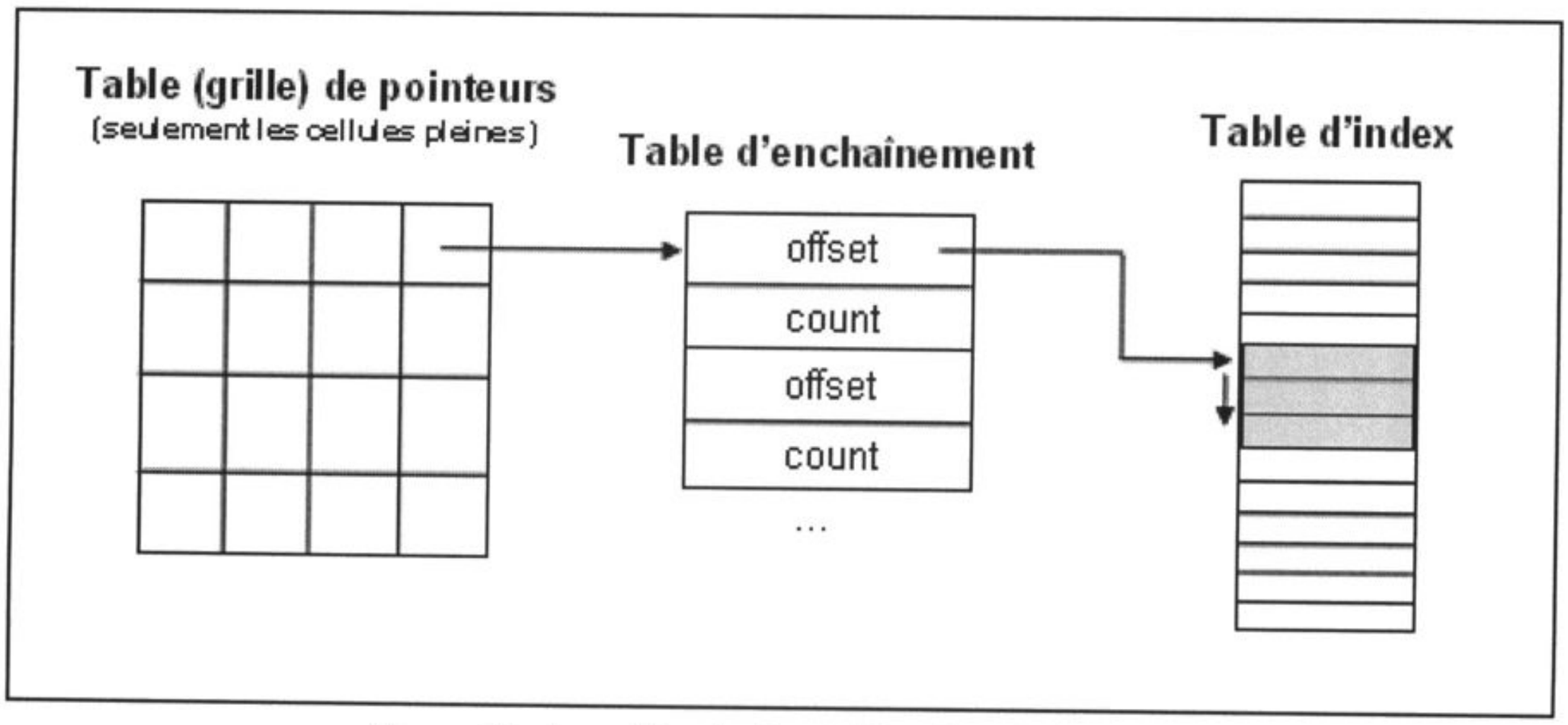


Figure 19 : La méthode des tables d'enchaînement

4.3 Travail à effectuer

Afin d'évaluer les performances des méthodes sur un cycle complet de prototypage, nous avons décidé d'effectuer une manipulation simple mais courante en prototypage : charger les données, générer une vue et décharger les données. La plupart des algorithmes utilisés utilisent le mécanisme de snapshots pour accélérer les traitements, car ils parcourent l'image entière. Comme le temps d'accès à une vue demeure constant (voir les snaphots) nous avons pu éliminer la nécessité de choisir un algorithme de traitement. De plus, nous avons écarté la possibilité d'évaluer les performances en écriture des méthodes, car les résultats seraient trop influencés par la manière dont les données pourraient être écrites. Plus précisément, il existe différentes manières d'effectuer l'écriture. Il y a par exemple l'écriture directe qui oblige l'utilisateur à posséder les clefs de la base de données (ce qui n'est pas recommandé) et il y a l'écriture tampon qui serait probablement la plus appropriée, mais qui introduit des délais non-désirables lorsqu'on veut mesurer l'efficacité de l'index comme tel. En lecture, par contre, le résultat est toujours identique, c'est-à-dire que nous obtenons une vue identique d'une méthode à l'autre. Cette propriété nous permet de comparer les méthodes efficacement et c'est pourquoi nous avons opté pour des tests qui génèrent des requêtes en lecture seulement.

Nous ferons varier la résolution de la vue à obtenir pour mesurer les forces et les faiblesses des différentes méthodes. En d'autres mots, la quantité de données dans la base de données sera fixe, c'est la requête d'information qui va changer. Plus la résolution demandée sera élevée, plus les requêtes seront précises et nombreuses.

Nous avons choisi trois images de tests pour représenter trois situations typiques en télédétection : l'image matricielle traditionnelle (aérienne), l'image multi-résolution (satellite) et l'image ponctuelle (lidar).

Image matricielle (pan only.mb)

Cette image est la bande panchromatique d'une image satellite. Elle comporte une seule résolution et un seul canal de données. Elle est facilement représentée dans une matrice.

Figure 20 : Image de test : matricielle

Image multi-résolution (ms-pan.mb)

Cette image est une photographie satellite comportant une bande panchromatique et une bande multi-spectrale. La bande panchromatique est de meilleure résolution que l'autre. On peut la représenter comme une série de matrices de tailles différentes, chaque matrice représentant un canal de données. L'image ci-dessous est une vue fusionnée des différentes représentations présentée sous la forme d'une image couleur. Dans ce cas, il s'agit d'une opération de pansharpening, une technique décrite plus tôt.

Figure 21 : Image de test : multi-résolution (*pansharpen*)

Image ponctuelle (lidar_crop.mb)

Cette image est un scan lidar pur. Elle ne comporte que des données ponctuelles et pourrait être représentée par un nuage de points. L'image ci-dessous est une vue générée à partir des données ponctuelles.

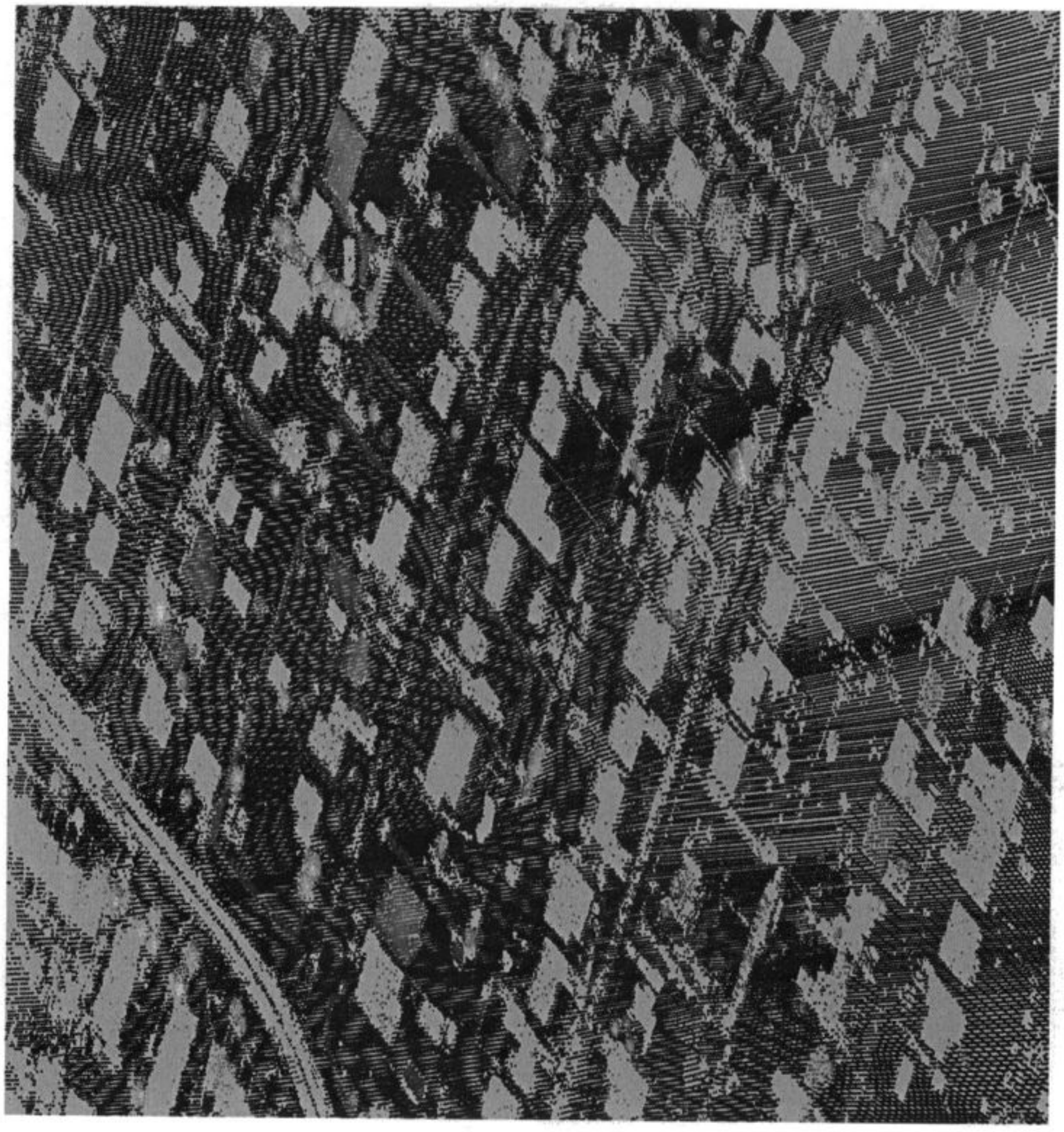


Figure 22 : Image de test : ponctuelle

4.4 Mesures de performance

Les mesures de performance choisies pour évaluer les méthodes sont :

Nombre de cellules (charge) : mesure de la charge soumise. Il s'agit de la résolution demandée et sera mesurée en MégaPixels.

Note au sujet de l'utilisation du terme MégaPixels : En télédétection, la résolution est plus souvent exprimée en mètres/pixel qui en retour définit le nombre de cellules spatiales possibles. Comme les trois différents fichiers ne couvrent pas la même superficie ET que la résolution n'est pas toujours homogène (considérons le cas de l'image LIDAR), il deviendrait difficile de comparer les graphiques entre eux. C'est pourquoi nous avons choisi une mesure plus générale et indépendante du fichier à traiter : le nombre de cellules spatiales à représenter. Comme chaque cellule spatiale fait référence à un pixel à l'écran, nous avons choisi le terme MégaPixel. 15 MégaPixels correspond à une image 3872x3872 pixels à l'écran.

Temps de chargement : évalue la rapidité à laquelle l'index spatial peut s'installer dans un contexte donné.

Temps de travail : temps qu'a pris le noyau à exécuter les requêtes.

Mémoire utilisée : quantité de mémoire requise pour représenter les données. Une méthode qui utilise trop de mémoire aura tendance à se saturer rapidement.

Temps de déchargement ou de nettoyage : évalue l'efficacité de l'index spatial à libérer les données qu'il exploitait. La lenteur est un signe de fragmentation.

4.5 Bancs d'essai

La collecte des résultats s'est vite avérée fastidieuse et il en a fallu de peu pour nous motiver à construire un banc d'essai automatisé. Une fois le banc d'essai réalisé, il a simplement fallu indiquer quels tests il fallait effectuer et laisser l'ordinateur travailler pendant quelques heures…

Le banc d'essai a été réalisé rapidement sous la forme de boucles imbriquées et de listes de paramètres à tester. À même le code, nous inscrivions les paramètres à exécuter dans les listes sous la forme suivante :

```
QList<Method> methods;
methods.push_back(Method::LookupTables);
methods.push_back(Method::BucketGrid);
methods.push_back(Method::Grid);
methods.push_back(Method::KDTree);
methods.push_back(Method::MXQuadTree);
methods.push_back(Method::RTree);

QList<QString> files;
files.push_back(base_path + "lidar_crop.mb");
files.push_back(base_path + "pan only.mb");
files.push_back(base_path + "ms-pan.mb");

QList<double> scales;
scales.push_back(500.0);
scales.push_back(250.0);
scales.push_back(100.0);
scales.push_back(75.0);
scales.push_back(50.0);
scales.push_back(25.0);
scales.push_back(15.0);
scales.push_back(10.0);
```

Des boucles imbriquées exécutaient alors chaque combinaison de tests. L'ordre des boucles a été déterminé d'avance pour regrouper les résultats d'une manière logique.

4.6 Équipement utilisé

Nous avons effectué les tests sur deux ordinateurs dont voici les caractéristiques principales :

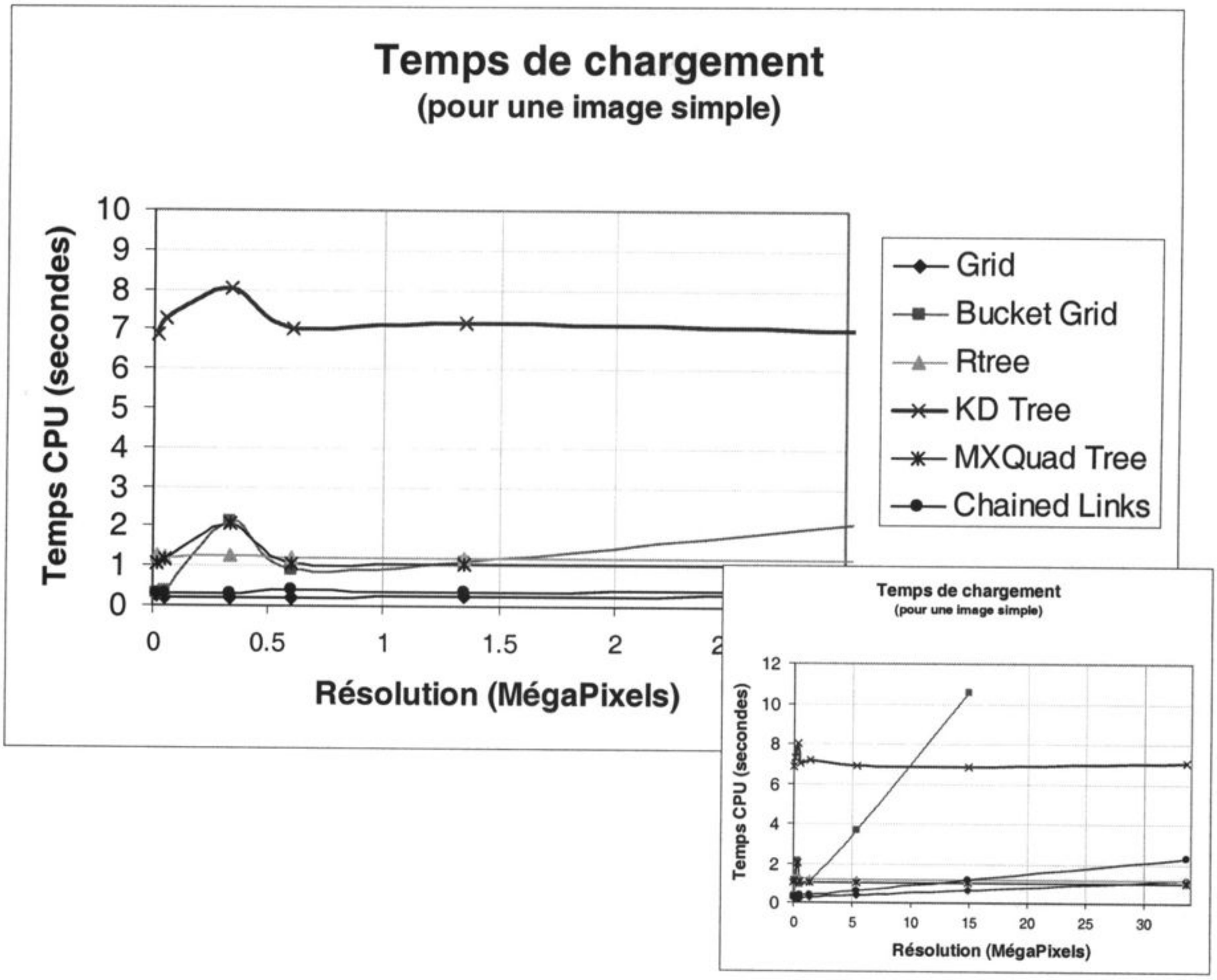
Ordinateur	Processeur	Mémoire Vive	Disque dur
#1	AMD Athlon 64 3700+ (2.22GHz)	2 Gb	Seagate SATA 7200.9 300 Gb
#2	Intel Centrino 1.6 GHz	1 Gb	IBM Hitachi 7200 40 Gb

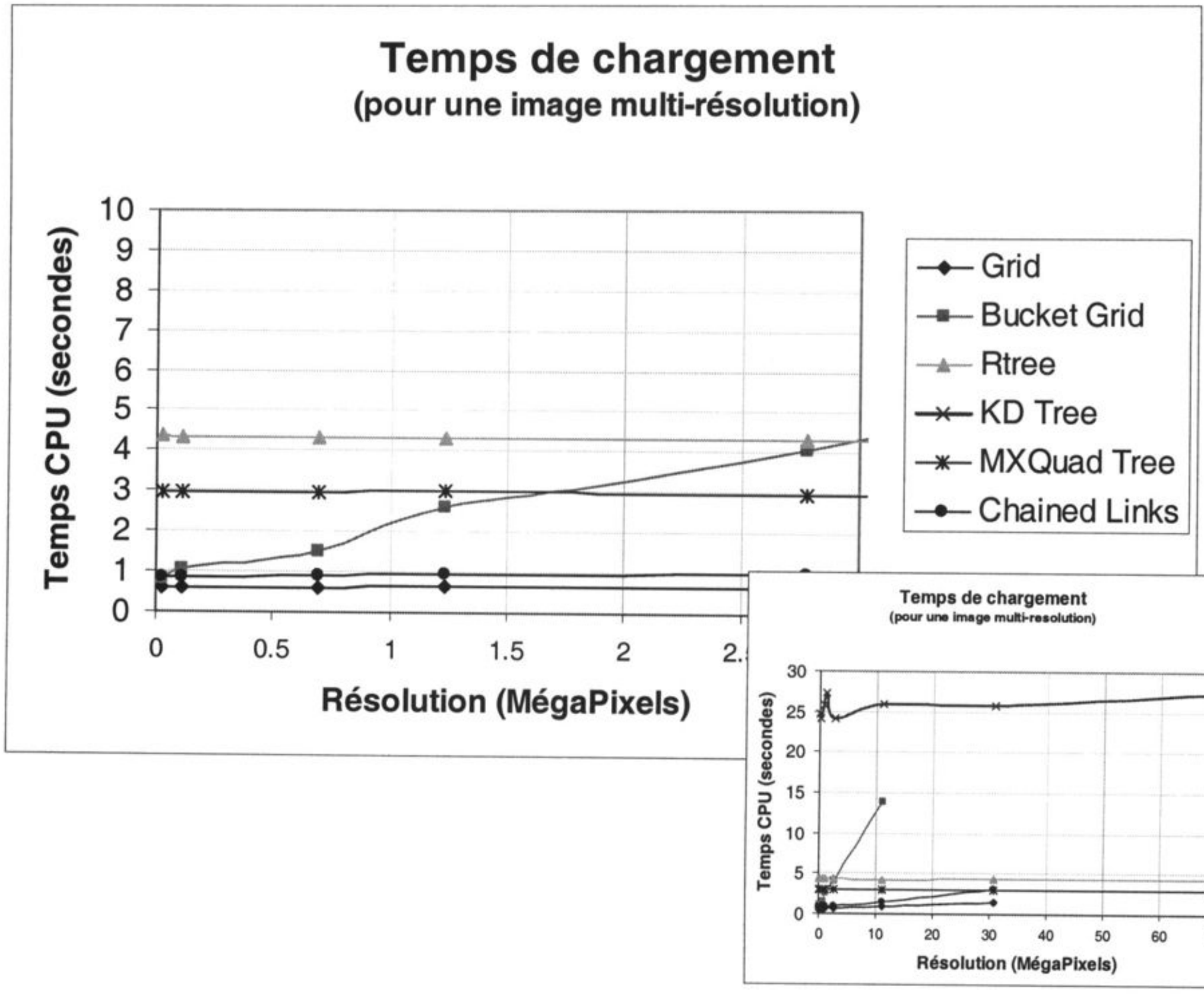
Les résultats présentés sont ceux obtenus avec le #1. Les résultats obtenus avec le #2 présentent les mêmes conclusions et ont été omises pour alléger la présentation. Vous trouverez les données brutes provenant des deux ordinateurs en annexe.

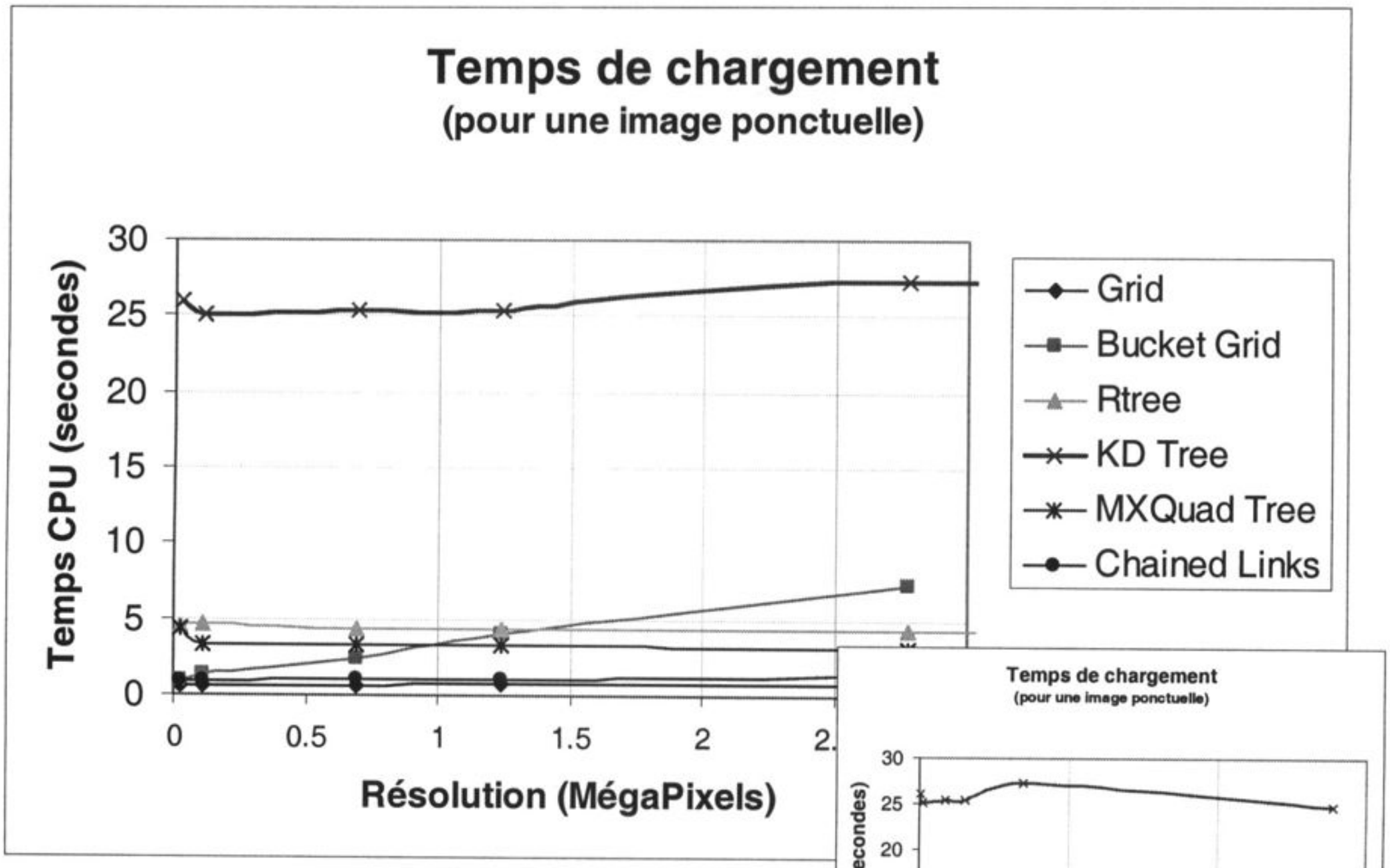
Du point de vue logiciel, nous avons utilisé la bibliothèque **Qt de Trolltech**. Il s'agit d'une bibliothèque de classes qui modélise tout ce dont un programmeur pourrait avoir besoin, en incluant les interfaces graphiques. Le principal avantage est que nous créons alors des programmes portables sur linux, windows et mac sans effort. Cette bibliothèque simplifie réellement le travail de la programmation en offrant un support de travail robuste et flexible. La compagnie Trolltech met à la disposition de tous une version *Open Source* tant que le développement des logiciels se fait sans but lucratif. Le choix de cette bibliothèque comme fondation à la plateforme Feature Ex-RA était en soi un grand en avant quant au respect des bonnes pratiques de génie logiciel.

5 Résultats

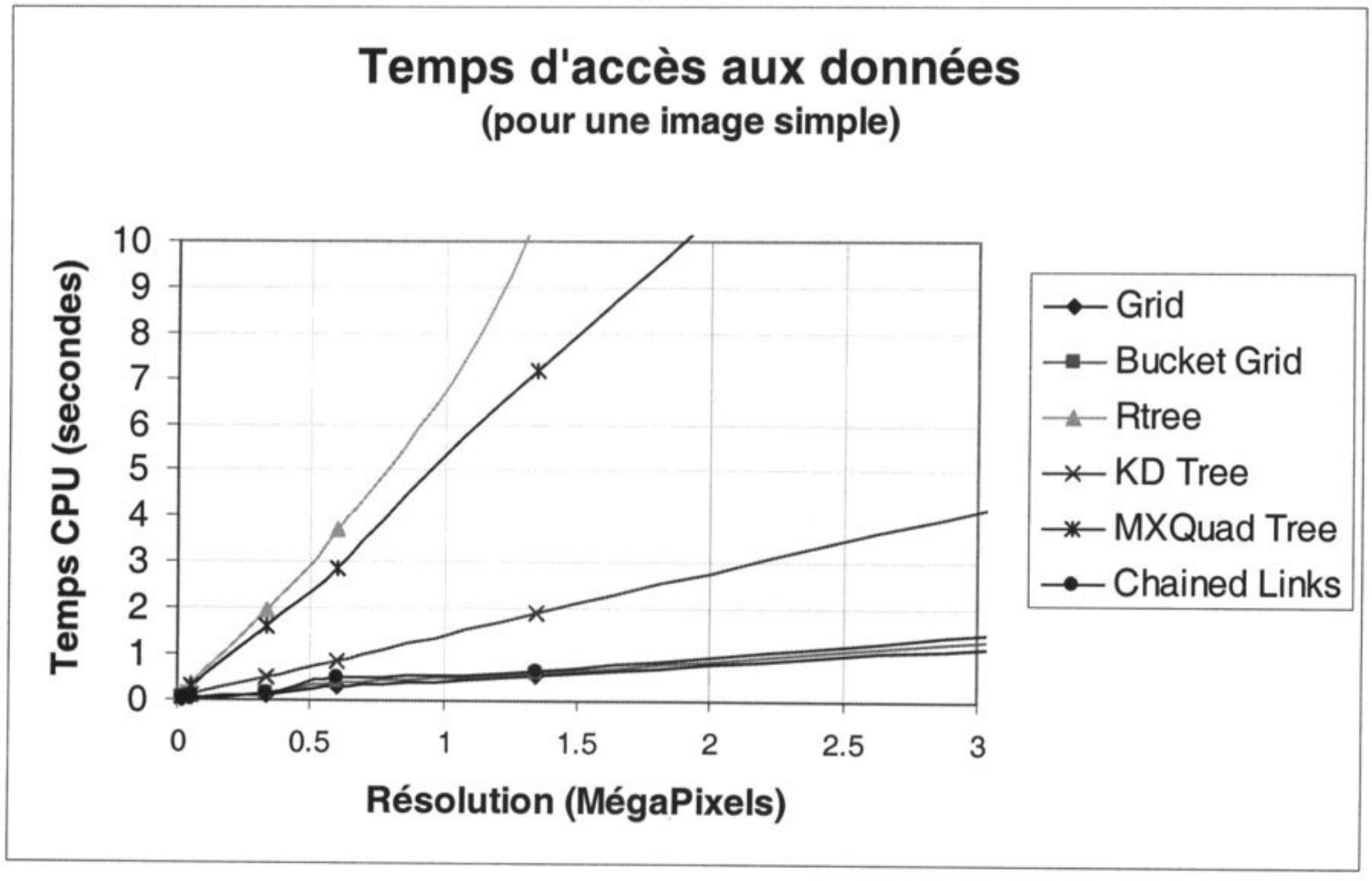
5.1 Temps de Chargement

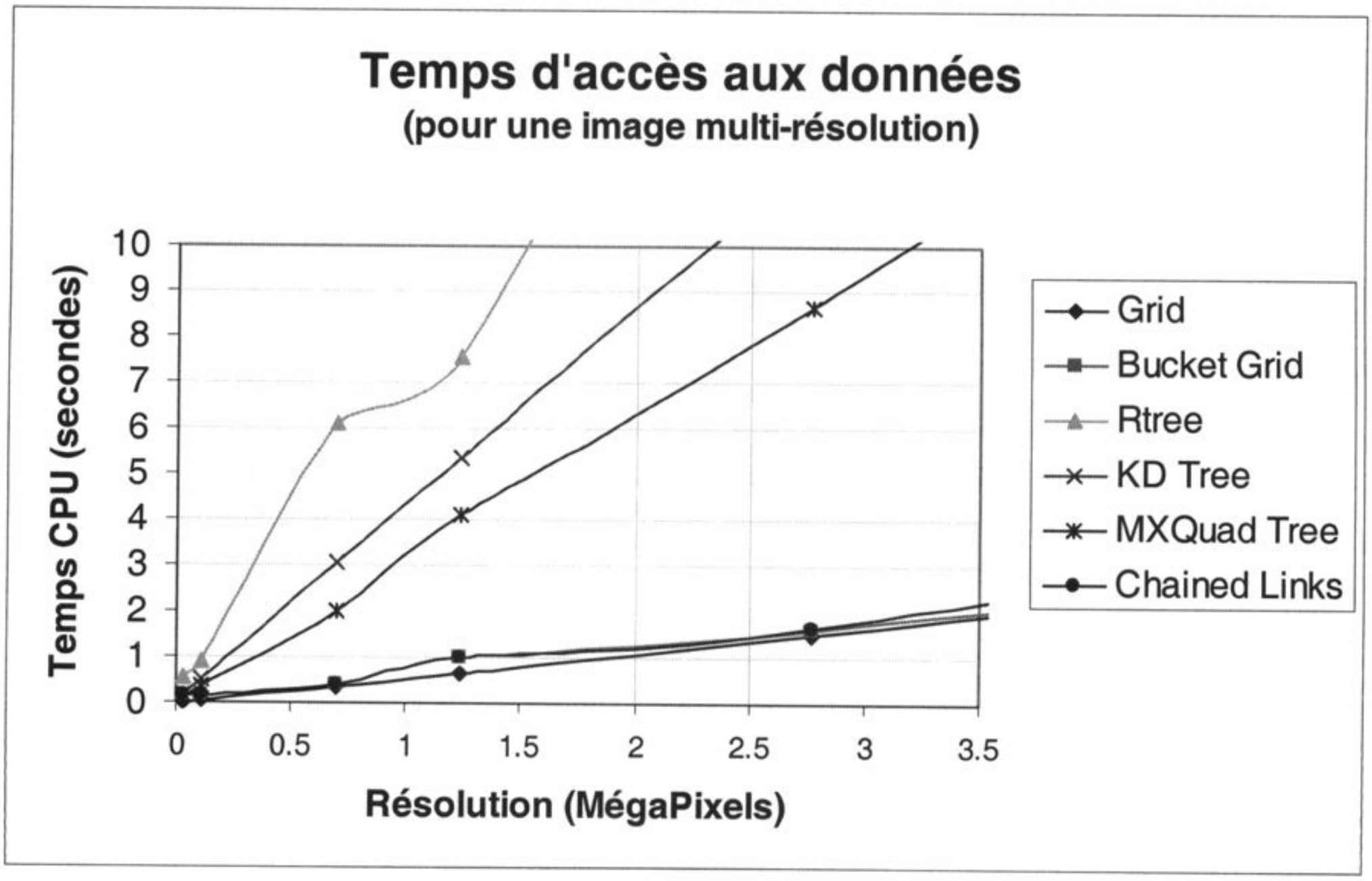


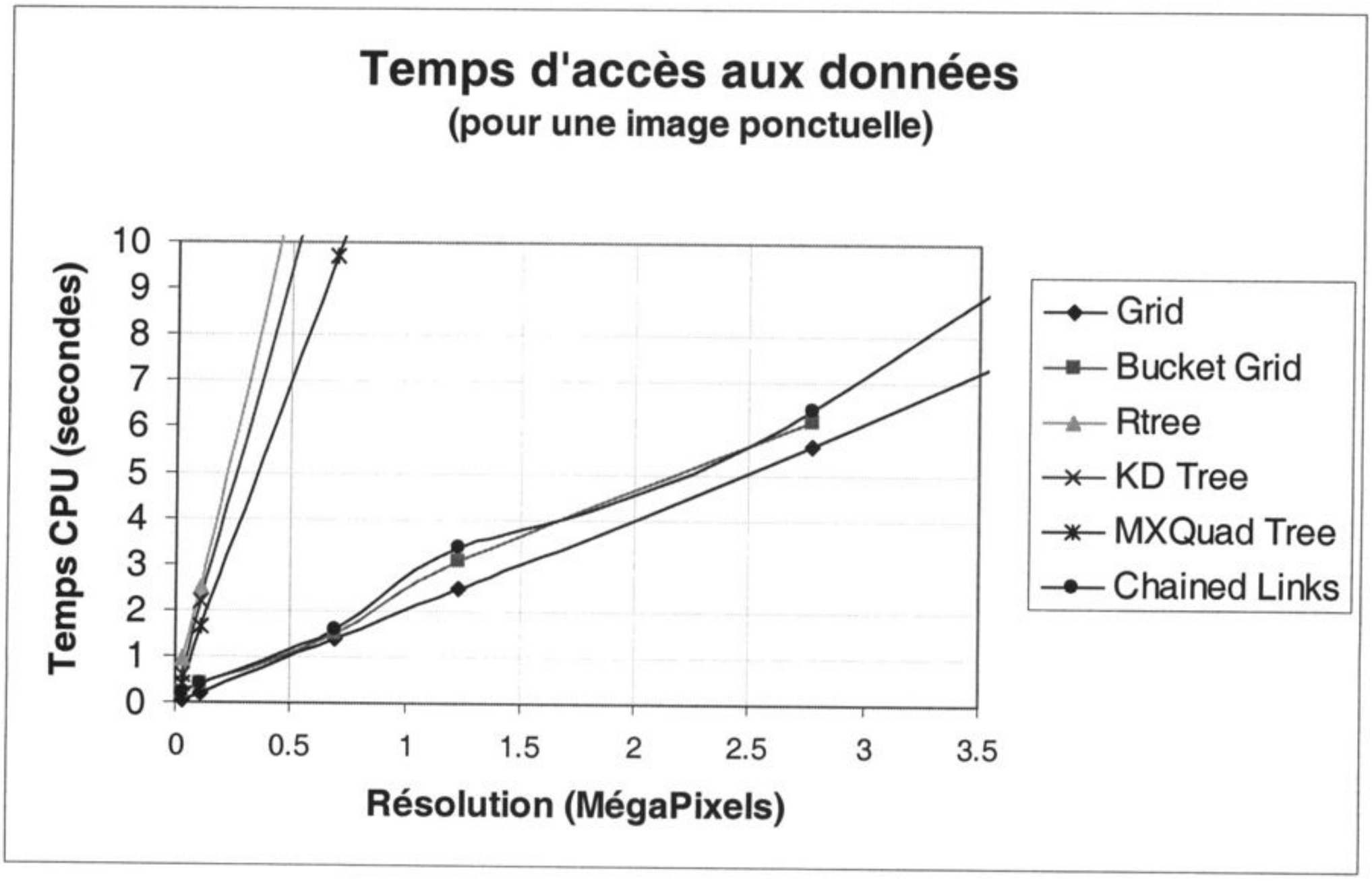

Temps de chargement
(pour une image multi-résolution)
Temps CPU (secondes)
0
1
2
3
4
5
6
7
8
9
10
0
0.5
1
1.5
2
Résolution (MégaPixels)
Grid
Bucket Grid
Rtree
KD Tree
MXQuad Tree
Chained Links
Temps de chargement
(pour une image multi-resolution)
Temps CPU (secondes)
0
5
10
15
20
25
30
0
10
20
30
40
50
60
70
Résolution (MégaPixels)


Temps de chargement
(pour une image ponctuelle)
Temps CPU (secondes)
30
25
20
15
10
5
0
0
0.5
1
1.5
2
Résolution (MégaPixels)
Grid
Bucket Grid
Rtree
KD Tree
MXQuad Tree
Chained Links
Temps de chargement
(pour une image ponctuelle)
Temps CPU (secondes)
30
25
20
15
10
5
0
0
2
4
6
8
10
12
Résolution (MégaPixels)

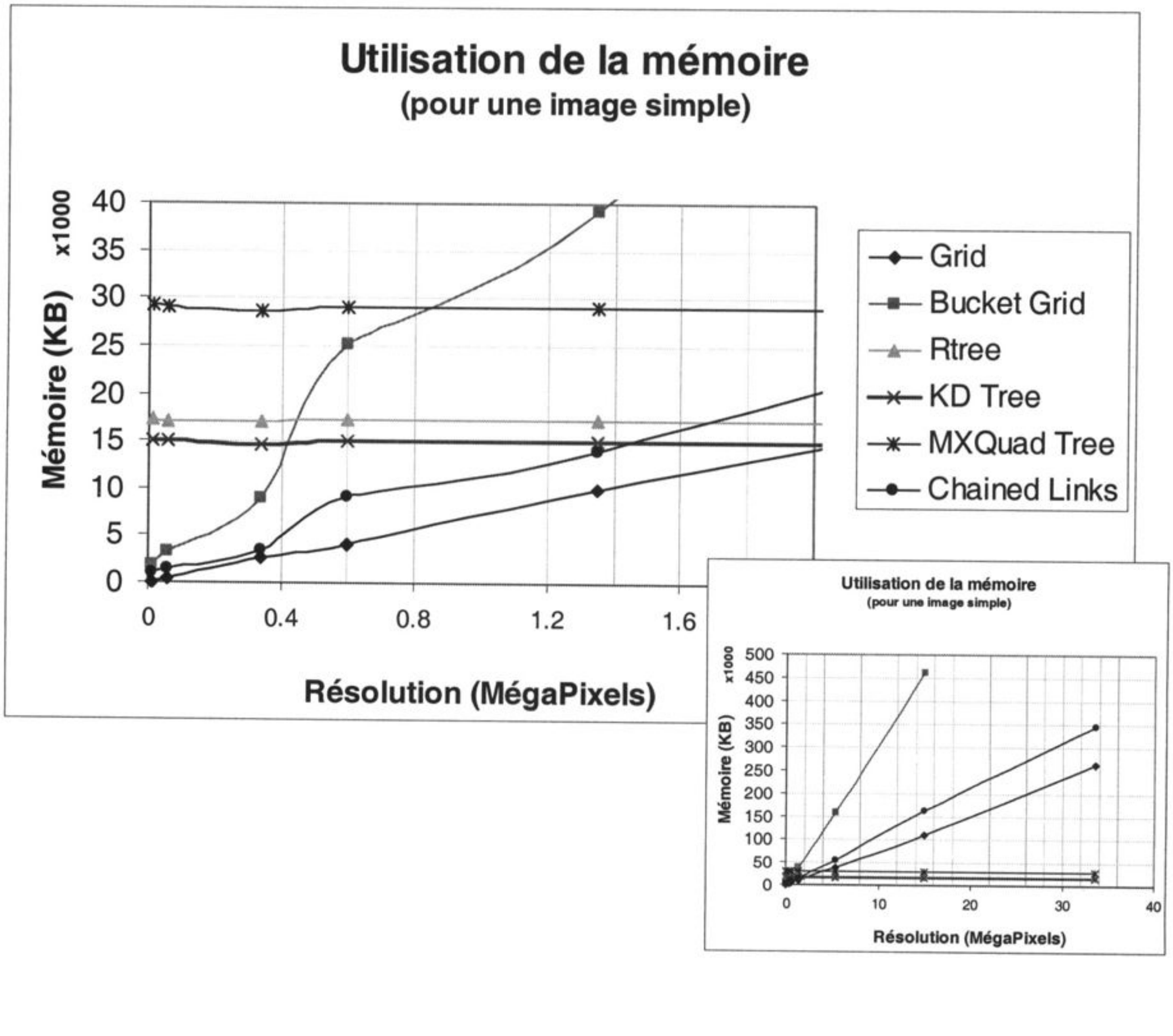
5.2 Temps de travail

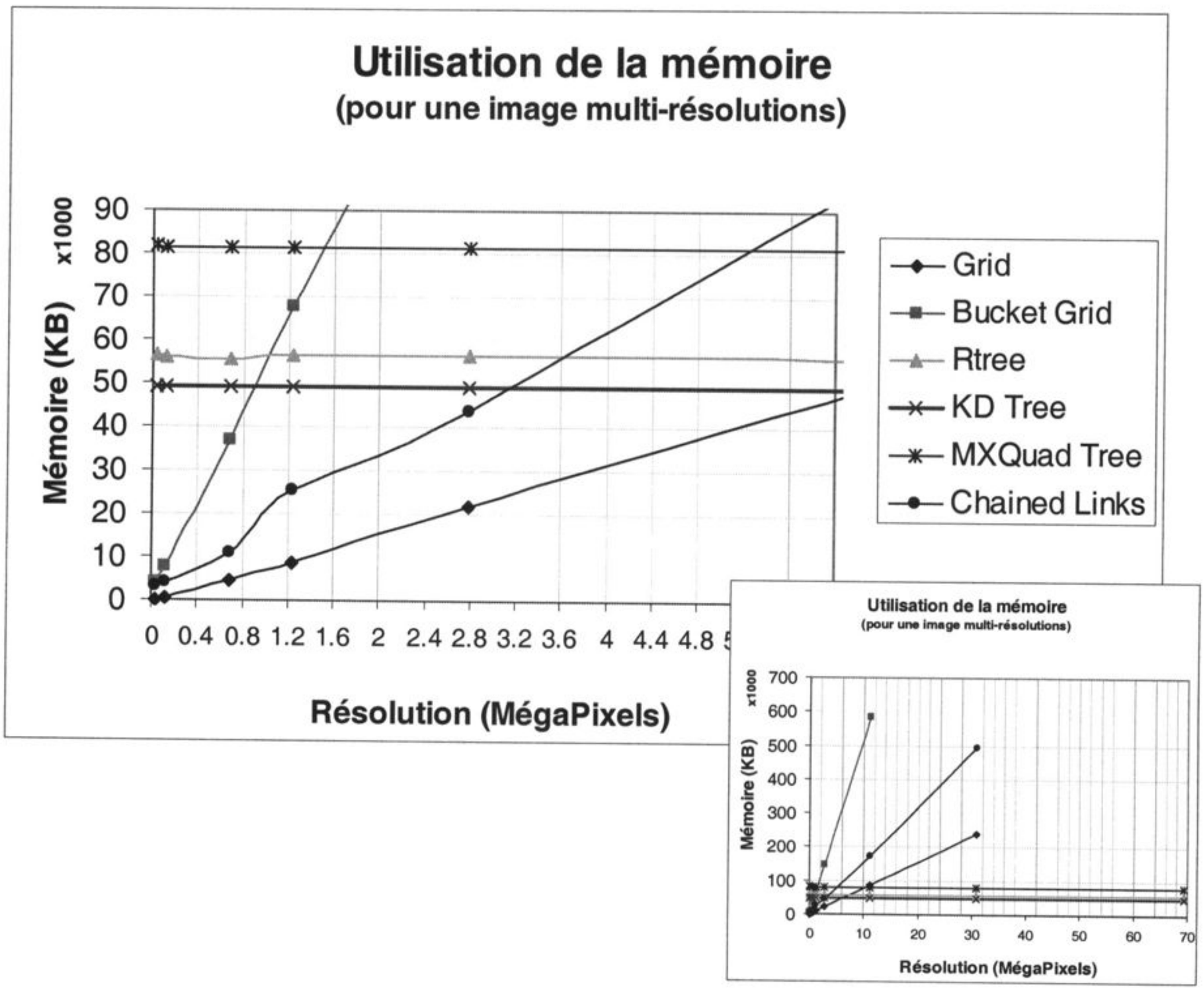


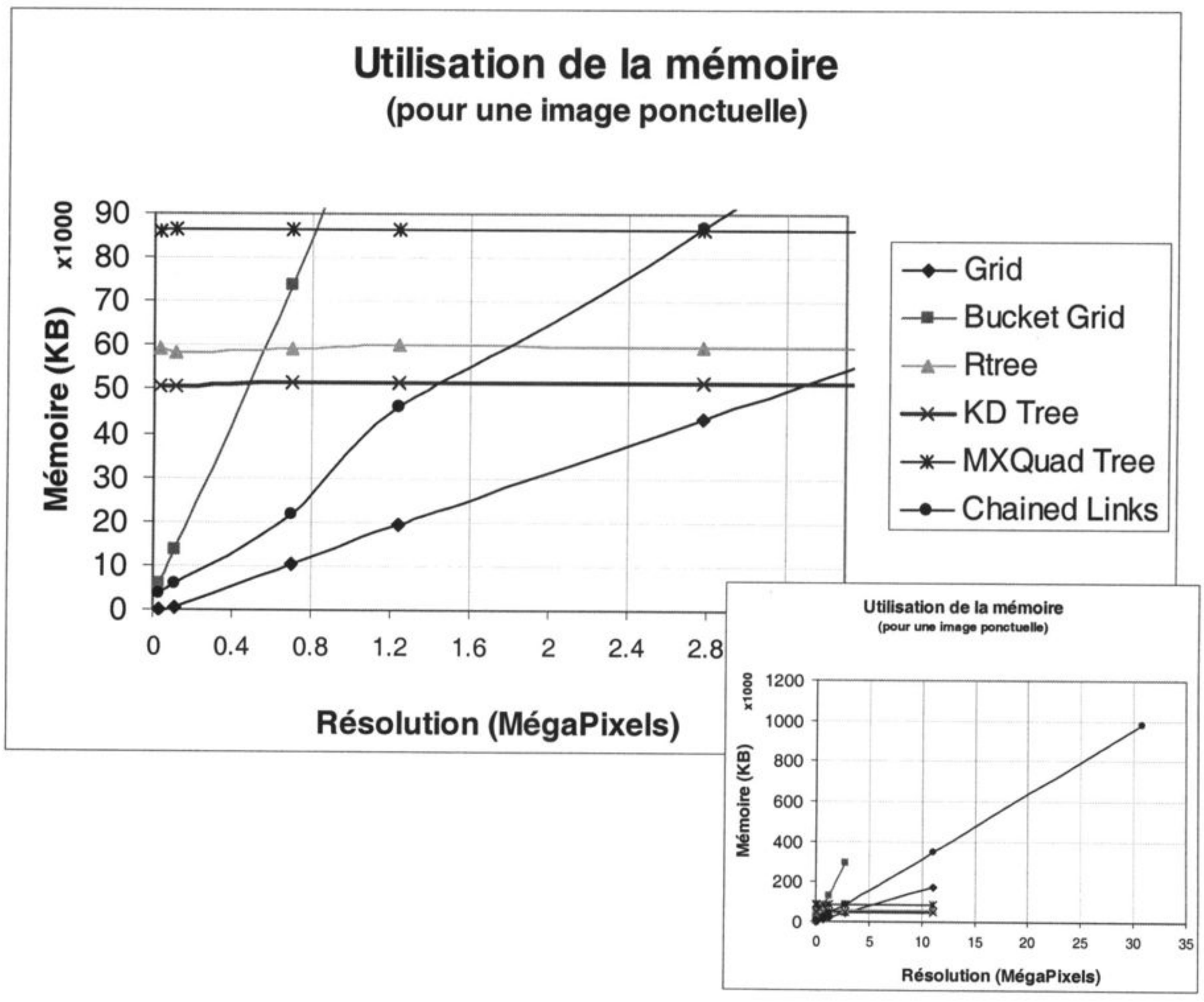



Temps d'accès aux données
(pour une image ponctuelle)
Temps CPU (secondes)
0
1
2
3
4
5
6
7
8
9
10
0
0.5
1
1.5
2
2.5
3
3.5
Résolution (MégaPixels)
Grid
Bucket Grid
Rtree
KD Tree
MXQuad Tree
Chained Links

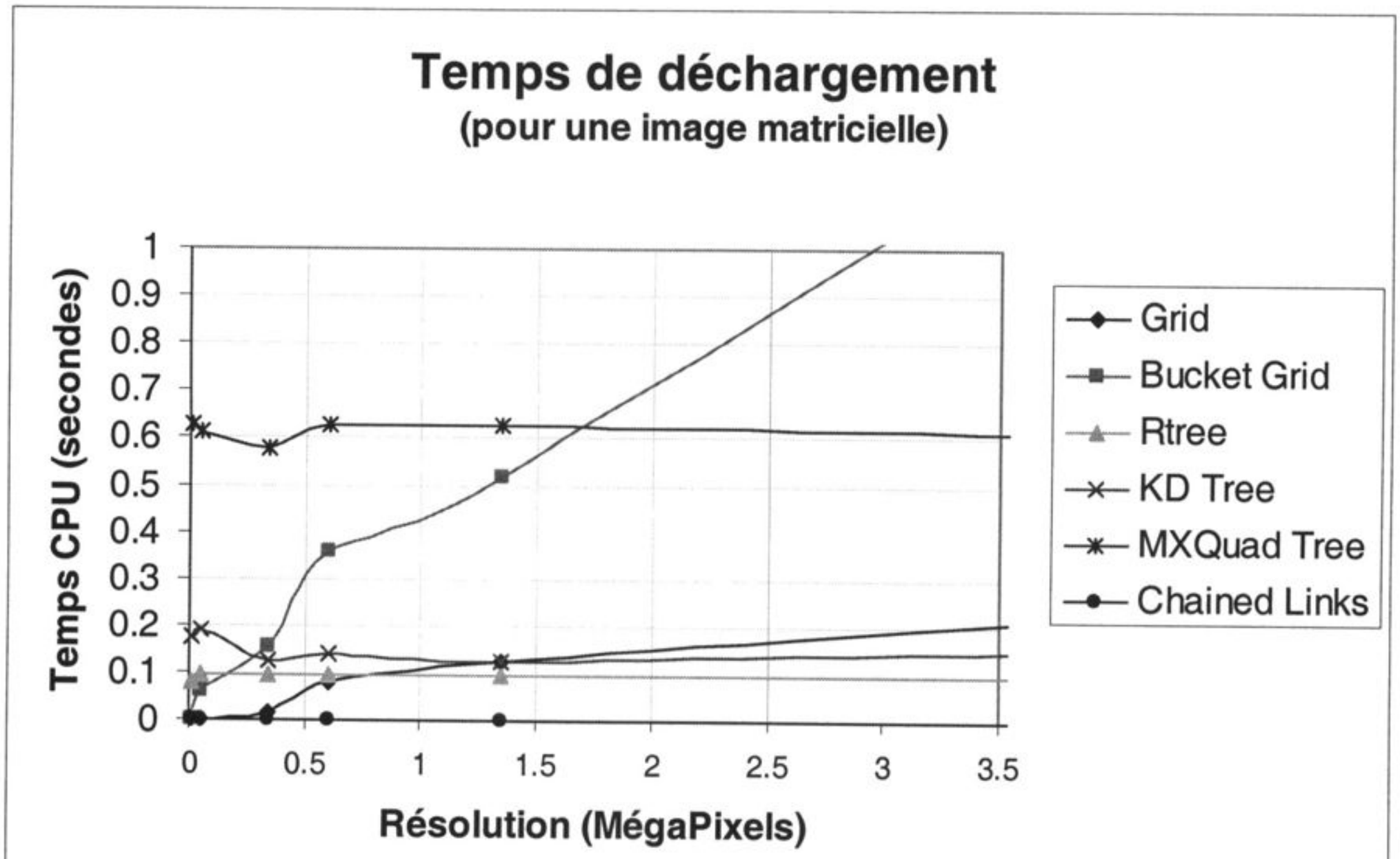
5.3 Mémoire utilisée

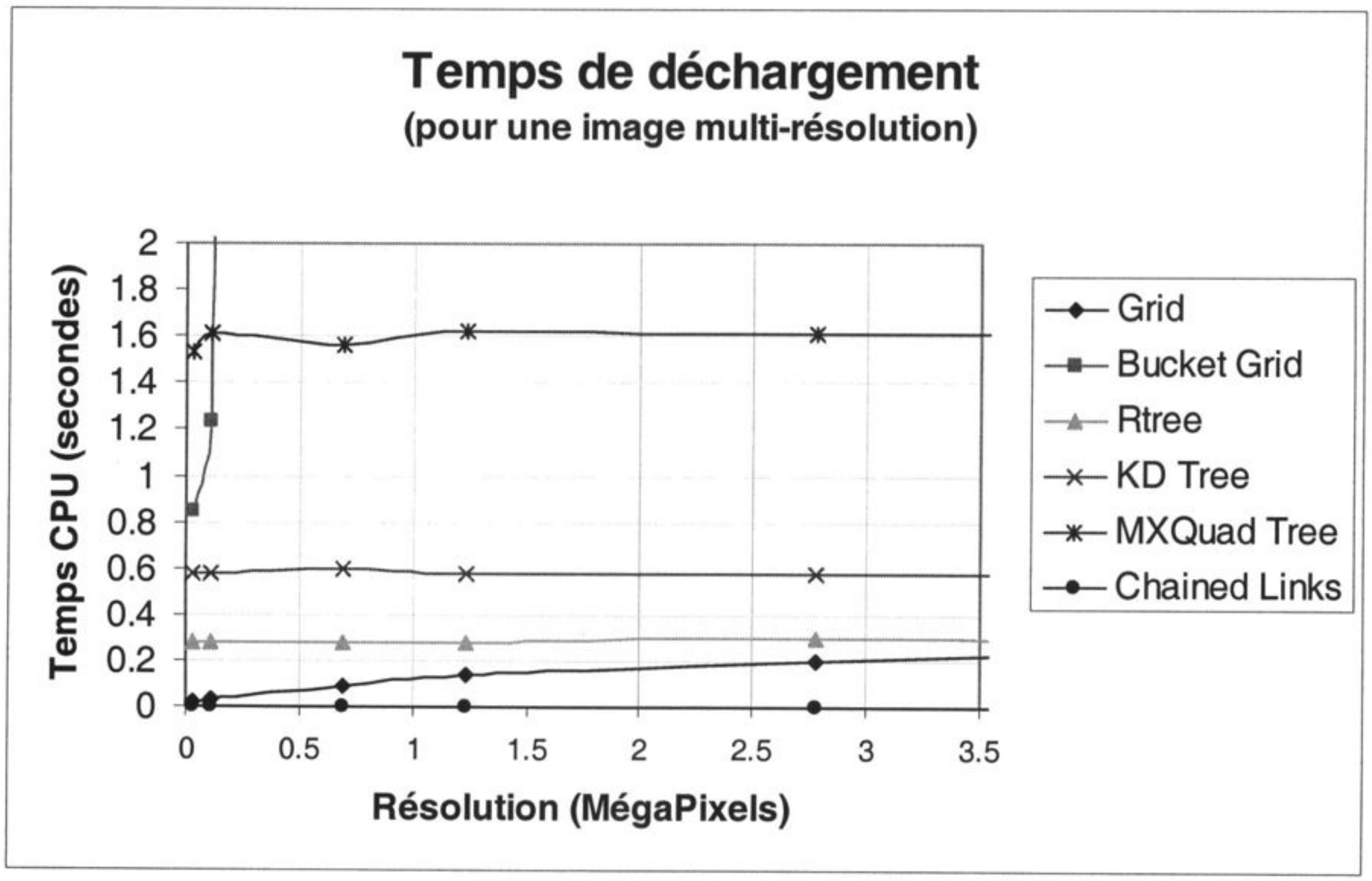


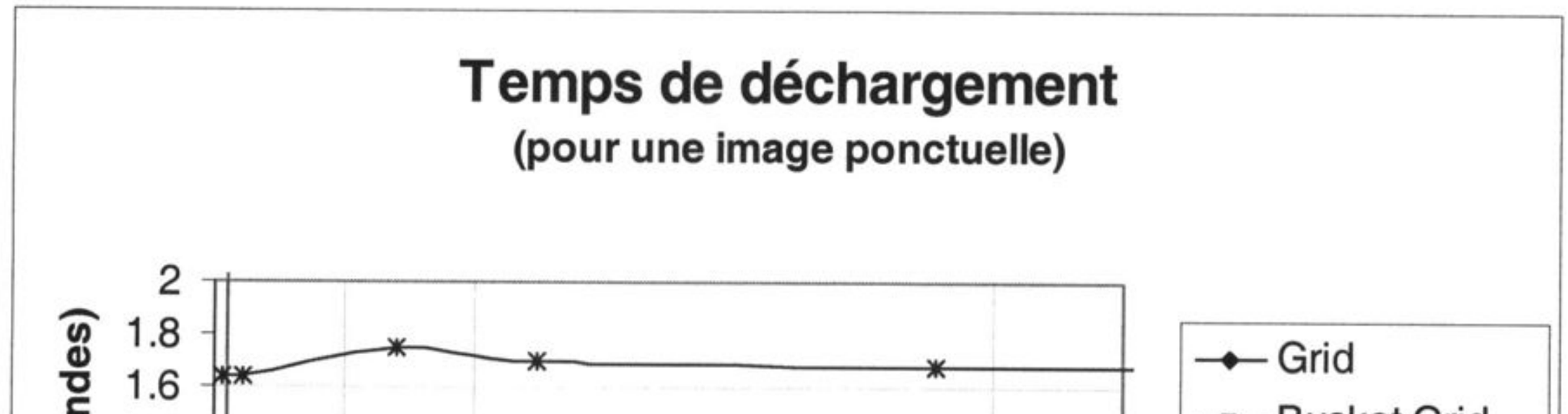

Utilisation de la mémoire
(pour une image multi-résolutions)
Mémoire (KB) x1000
90
80
70
60
50
40
30
20
10
0
0 0.4 0.8 1.2 1.6 2 2.4 2.8 3.2 3.6 4 4.4 4.8
Résolution (MégaPixels)
Grid
Bucket Grid
Rtree
KD Tree
MXQuad Tree
Chained Links
Utilisation de la mémoire
(pour une image multi-résolutions)
Mémoire (KB) x1000
700
600
500
400
300
200
100
0
0 10 20 30 40 50 60 70
Résolution (MégaPixels)


Utilisation de la mémoire
(pour une image ponctuelle)
Mémoire (KB) x1000
90
80
70
60
50
40
30
20
10
0
0
0.4
0.8
1.2
1.6
2
2.4
2.8
Résolution (MégaPixels)
Grid
Bucket Grid
Rtree
KD Tree
MXQuad Tree
Chained Links
Utilisation de la mémoire
(pour une image ponctuelle)
Mémoire (KB) x1000
1200
1000
800
600
400
200
0
0
5
10
15
20
25
30
35
Résolution (MégaPixels)

5.4 Temps de déchargement




Temps de déchargement
(pour une image multi-résolution)
Temps CPU (secondes)
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5
2
2.5
3
3.5
Résolution (MégaPixels)
Grid
Bucket Grid
Rtree
KD Tree
MXQuad Tree
Chained Links


Temps de déchargement
(pour une image ponctuelle)
Temps CPU (secondes)
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5
2
2.5
3
3.5
Résolution (MégaPixels)
Grid
Bucket Grid
Rtree
KD Tree
MXQuad Tree
Chained Links

6 Analyse des résultats

6.1 Analyse des performances des méthodes

6.1.1 Grille

La grille simple apparaît comme la méthode de choix pour le prototypage en télédétection. Il supplante ses concurrents sous tous les aspects mesurés sauf pour le temps de déchargement où la méthode d'enchaînements a l'avantage. Malheureusement, cette impression est fausse…

Nous l'avions laissé entendre dans la section méthodologie : la grille est un concurrent déloyal. En fait il ne produit pas des résultats identiques aux autres pour la simple et bonne raison qu'il ne supporte pas la possibilité de stocker plus d'un point dans une cellule donnée. Lorsqu'on augmente la résolution, la grille discrimine des données au lieu de les conserver comme le font toutes les autres méthodes.

Pourquoi alors la grille n'a pas été implémentée de façon à supporter plusieurs points par cellule? Parce qu'il s'agirait alors d'une grille à baquets!! Il faut donc comprendre que la grille présentée ici n'est qu'une référence : on compare les cinq autres méthodes à une technique simple de rastérisation.

Note : malgré sa simplicité, la grille simple obtient une cote de fragmentation (voir temps de déchargement) comparable aux arbres de recherche. La raison est due au fait que toute la mémoire est allouée dans un seul énorme bloc, ce qui implique une fragmentation due au fait que le système d'exploitation n'ait pas pu trouver un bloc entier disponible en mémoire vive.

6.1.2 Grille à baquets

La grille à baquets est en réalité une implémentation correcte de la grille simple comme méthode de fusion de représentations. En effet, une bonne implémentation répond aux requêtes du contenu des cellules par une liste d'index pointant aux données concernées dans la base de données. La grille était limitée au stockage d'une seule valeur, et donc ne remplissait pas correctement son rôle. La grille à baquets n'est qu'une adaptation de la grille simple où chaque cellule de la grille pointe vers un vecteur d'index.

Ce genre d'implémentation montre comment il n'est pas adapté au monde ponctuel ou multi-résolution. Tous les tests démontrent une saturation rapide de la mémoire. C'est dû en grande partie aux pertes causées par les « trous » présentés dans la section sur les index spatiaux.

Cependant, sa simplicité de réalisation en fait un bon candidat pour toute situation où moins de 1 Méga Pixels de données seront sollicitées.

6.1.3 L'arbre R

L' arbre R a répondu à nos attentes au niveau de l'utilisation de la mémoire : ne sont stockés que les index et un peu d'information quant à leur position. Il en résulte une utilisation stable et constante qui ne dépend pas de la résolution demandée, mais de la quantité de données à stocker. L'utilisation d'un arbre est approprié pour toute opération nécessitant plus d'un Méga Pixel.

Sa principale faiblesse se situe au niveau du temps d'exécution des requêtes : sa tendance devient rapidement exponentielle en fonction de la résolution demandée. C'est dû au fait que les requêtes demandent une région spatiale de plus en plus petite, nécessitant des calculs de proximité de plus en plus nombreux. Une optimisation de la fonction de proximité apporterait de grandes améliorations aux performances de l'index spatial.

De plus, l'arbre R est l'arbre le plus complexe des trois à réaliser. Le rendant plus ou moins intéressant vis-à-vis les deux autres arbres.

6.1.4 L'arbre k-d

L'arbre k-d est l'arbre le plus simple à réaliser. Il est aussi le seul arbre pouvant être adapté facilement à un espace de plus de deux dimensions. En effet, les eux autres arbres sont conçus uniquement pour un espace bidimensionnel.

Il offre non seulement la plus basse consommation en mémoire pour la catégorie des arbres de recherche, mais ses performances en termes de temps CPU sont aussi meilleures. La consommation CPU est linéaire et proportionnelle à la résolution. Peu importe la région spatiale sollicitée, le temps nécessaire à la localisation de l'information est toujours régulier. De plus, les points qui sont à proximité spatialement le sont aussi dans l'arbre de recherche.

La plus grande faiblesse de cette méthode est le temps de chargement. Son temps de chargement est de loin supérieur à tous les autres. Une différence d'au moins 500% !! En minimisant les chargements, un chercheur pourrait en tirer partie efficacement.

6.1.5 L'arbre quaternaire matriciel

L'arbre quaternaire matriciel est un arbre qui s'est classé entre les deux premiers : il est un peu plus compliqué à réaliser que l'arbre k-d, sa consommation en CPU est raisonnable, mais il présente la pire consommation mémoire des trois.

Cet arbre permet de repérer efficacement de l'information bidimensionnelle, mais nécessite la création de nœuds de structure. Dans cet arbre, les données sont stockées aux feuilles, et plus la résolution demandée sera grande, plus le nombre de nœuds augmentera, car ce sont les nœuds qui créent la structure matricielle de l'arbre quaternaire matriciel.

L'arbre quaternaire matriciel a aussi la faiblesse d'être plus adapté pour les surfaces carrées que d'autres. En effet, la résolution demandée est limitée par un multiple de 2 dans chaque direction. Souvenons-nous : l'arbre divise l'espace en quatre (en deux sur chaque dimension) à chaque niveau de résolution. On choisit le niveau de résolution et non pas la résolution à proprement parler. C'est un désavantage, car tous les autres offrent la possibilité de travailler à des résolutions arbitraires.

6.1.6 Tables d'enchaînements

La méthode par enchaînage est un mélange d'un arbre de recherche et de tables de recherche. Sa performance a été intéressante et est détaillée ci-dessous :

Du point de vue de l'utilisation de la mémoire, cette méthode apporte une nette amélioration à la grille à baquets (au moins 65% d'amélioration!) mais est dominée par les arbres de recherche à environ 3 MégaPixels selon le cas.

Du point de vue du temps de chargement, elle offre la meilleure performance jusqu'à 15 MégaPixels.

Du point de vue du temps d'accès, elle conserve les mêmes performances offertes par la grille à baquets. Performance qui domine toutes les méthodes basées sur les arbres de recherche.

Quant au temps de déchargement, la méthode par enchaînements présente la meilleure performance de toutes les méthodes, démontrant une très faible fragmentation de la mémoire. Comme nous l'avons expliqué dans la section des index spatiaux, cette méthode visait à pallier les graves problèmes de fragmentation que présentait la méthode de la grille à baquets.

6.2 Tableau comparatif

Voici un tableau récapitulatif des forces et faiblesses des différentes méthodes. La grille est représentée ici à titre de référence, mais il ne faut pas oublier qu'il est disqualifié à cause du fait qu'il ne remplit pas pleinement ses fonctions. Voir les analyses à ce sujet. Les méthodes ont été évaluées sur une échelle de 1 à 4 par rapport aux différentes qualités recherchées.

Explication des qualités recherchées et des symboles utilisés pour l'évaluation.

Qualité	Excellent (4)	Bon (3)	Pauvre (2)	Nul (1)
Chargement	Très court	Efficace	Tardif	Trop long
Mémoire	Économe	Dans les normes	Gourmand	Saturation trop rapide de la mémoire
Accès	Très rapide	Acceptable	Lent	Très lent
Fragmentation	Aucune	Moyenne	Élevée	Difficile à gérer
Simplicité	Simple	Acceptable	Compliqué	Trop compliqué

Méthode	Chargement	Mémoire	Accès	Fragmentation	Simplicité
Grille	4	2	4	3	4
Grille à baquets	1	1	4	1	4
Arbre R	3	3	1	3	2
Arbre quaternaire matriciel	3	3	3	3	3
Arbre k-d	2	4	3	3	3
Tables d'enchaînements	4	2	4	4	3

6.3 Choix de la meilleure méthode

D'après nos résultats et nos analyses, voici comment nous classerions les différentes méthodes. Parmi les meilleures, deux méthodes se démarquent des autres : l'arbre KD et la méthode des tables d'enchaînements. Les deux offrent un équilibre intéressant entre les différentes qualités recherchées. L'arbre KD trouve sa force au fur et au mesure que la quantité de données augmente alors que la méthode des tables d'enchaînements est la plus efficace lorsqu'il s'agit de plus petites quantités d'information.

Ensuite vient l'arbre quaternaire matriciel qui est dans une classe à part, intermédiaire à tous les niveaux. Il ne présente pas vraiment d'avantages distinctifs, mais reste tout de même dans les normes. Son temps d'accès moyen n'en fait pas une méthode idéale.

En dernier viennent l'arbre R et la grille à baquets qui présentent de sérieuses lacunes. L'arbre R d'abord pour son temps d'accès médiocre, et la grille à baquets pour ses problèmes de saturation de la mémoire.

L'arbre k-d ferait un excellent outil de travail d'autant plus qu'il peut être utilisé à plus de deux dimensions, mais en tenant compte du contexte et des critères définis la méthode par tables d'enchaînements serait favorable à une telle plateforme de prototypage. Tout d'abord parce qu'elle offre parmi les meilleurs temps d'accès, mais aussi parce qu'elle est la plus efficace en tous points pour des données impliquant moins de 15 MégaPixels d'information.

7 Conclusion

Ce mémoire était centré sur deux objectifs principaux : la réalisation d'une plateforme de prototypage pour la télédétection et l'évaluation du choix d'un index spatial sur les performances du système.

Nous avons d'abord réalisé la plateforme Feature Ex-RA et expliqué comment elle respectait de bons principes de génie logiciel. L'utilisation de plugins pour le chargement et le traitement de données la rend flexible et attrayante pour l'usager. Les plugins de chargement évitent les tracas de la conversion de données alors que les plugins de traitement encapsulent les algorithmes dans un cadre générique qui leur permette d'être interchangés et enchaînés facilement dans une chaîne de traitement. Le système de vues et de snapshots rend la fusion de représentations transparente pour l'utilisateur et les algorithmes, sans pour autant lui enlever sa flexibilité, car l'index spatial peut être choisi ou remplacé facilement. Finalement, un banc d'essai a été réalisé pour faciliter l'étape de la collecte d'informations.

Parallèlement, nous avons implémenté cinq méthodes différentes à utiliser dans l'index spatial. Nous avons vu et compris pourquoi la grille n'est pas adaptée à ce genre de travail et c'est pourquoi il a été écarté du choix de la meilleure méthode. Nous avons aussi réalisé une adaptation de la grille à baquets que nous avons nommée tables d'enchaînements (*Chained Links*).

Cette dernière s'est avérée très efficace durant les tests à un tel point qu'elle a été choisie comme méthode idéale pour la plateforme de prototypage. L'arbre KD a terminé ex-æquo et est recommandé pour des volumes de données plus élevés.

La prochaine étape consisterait à implanter un système transactionnel de gestion des requêtes. Il y aurait aussi moyen d'améliorer la performance mémoire de la méthode des tables d'enchaînements en changeant la table maîtresse pour une table de hachage.

Résultats Bruts

A1. Ordinateur de bureau Athlon 64

Mémoire utilisée

Mémoire utilisée Load (cells)	Image Matricielle Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
13872	88	1872	17256	15112	29204	1020
54742	420	3204	17136	15096	29000	1440
338348	2616	9012	17136	14668	28692	3508
599864	3980	25196	17172	15044	29116	9152
1349694	9960	39116	17332	15060	29116	13912
5389384	38868	157128	17296	15060	29116	55564
14960810	111224	461632	17260	16044	29116	162148
33653456	263356		17436	15128	29116	346960

Mémoire Utilisée Load (cells)	Image Multi Résolutions Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
28208	136	4296	56380	48988	81644	3064
111818	324	7832	55940	48940	81488	4040
695072	4740	36600	55644	48968	81488	10888
1235703	8804	67540	56588	48972	81488	25392
2776953	21760	146416	56196	48948	81496	43480
11101145	86908	585484	55968	48968	81488	173800
30831008	239576		56148	48968	81488	495036
69357161			56024	48972	81488	

Mémoire Utilisée Load (cells)	Image Ponctuelle Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
28208	0	5840	59156	50728	85876	3568
111818	240	13652	58500	50768	86560	6028
695072	10188	73496	59108	51420	86568	21776
1235703	19364	132876	60112	51408	86560	46028
2776953	43496	293656	59720	51408	86588	86956
11101145	172776		59480	51416	86556	347592
30831008						980736
69357161						

Temps CPU au chargement

Loading Time — Image matricielle

Load (cells)	Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
13872	250	281	1265	6844	1047	297
54742	204	343	1235	7281	1172	296
338348	219	2125	1250	8016	2047	313
599864	218	922	1235	7000	1078	406
1349694	250	1110	1219	7172	1047	359
5389384	375	3687	1218	6891	1063	594
14960810	657	10563	1203	6828	1047	1188
33653456	1187		1203	7079	1047	2297

Loading Time — Image Multi-Résolutions

Load (cells)	Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
28208	594	828	4359	24625	2969	844
111818	593	1047	4312	24156	2985	843
695072	609	1531	4297	25797	2985	907
1235703	640	2610	4328	27250	3000	953
2776953	672	4047	4313	24156	2969	1015
11101145	937	13922	4250	25953	3015	1516
30831008	1562		4344	25766	3031	3078
69357161			4312	27297	3000	

Loading Time — Image Ponctuelle

Load (cells)	Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
28208	656	906	4546	26015	4313	906
111818	672	1391	4594	25078	3235	907
695072	672	2375	4390	25391	3234	984
1235703	719	4125	4391	25391	3250	1094
2776953	797	7391	4375	27281	3219	1297
11101145	1344		4312	24656	3234	2485
30831008						
69357161						

Temps CPU à l'exécution

Loading Time Load (cells)	Image matricielle Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
13872	0	47	172	31	63	47
54742	31	63	375	78	281	63
338348	125	156	1968	500	1625	156
599864	282	406	3703	859	2844	485
1349694	531	625	10640	1906	7172	641
5389384	2125	2422	80407	7343	26953	2656
14960810	6125	7016	412781	19969	67218	7671
33653456	14313		1679453	44750	167266	22000

Loading Time Load (cells)	Image Multi Resolution Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
28208	15	141	532	141	156	141
111818	63	172	922	516	375	172
695072	375	406	6125	3063	2015	406
1235703	657	984	7531	5375	4078	1016
2776953	1500	1625	23359	11907	8656	1641
11101145	6516	6234	136703	46141	35469	8812
30831008	17938		598734	126484	96391	31656
69357161			2281032	269890	225594	

Loading Time Load (cells)	Image Multi Resolution Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
28208	63	203	922	578	375	204
111818	218	391	2469	2172	1625	390
695072	1375	1485	15500	13094	9687	1578
1235703	2500	3062	26297	22984	16937	3359
2776953	5625	6172	78000	51344	38297	6407
11101145	24297		568172	202516	152031	35093
30831008						
69357161						

Temps de déchargement

Loading Time Load (cells)	Image matricielle Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
13872	0	0	78	172	625	0
54742	0	62	94	188	610	0
338348	16	156	94	125	578	0
599864	78	360	94	141	625	0
1349694	125	515	94	125	625	0
5389384	265	1937	93	157	609	0
14960810	500	6734	94	171	610	0
33653456	796		94	171	625	32

Loading Time Load (cells)	Image Multi Resolution Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
28208	16	844	281	578	1531	0
111818	31	1234	281	578	1609	0
695072	94	84828	281	593	1563	0
1235703	140	75781	282	578	1625	0
2776953	203	170359	297	578	1609	0
11101145	453	350594	234	578	1625	16
30831008	859		281	579	1578	32
69357161			265	563	1562	

Loading Time Load (cells)	Image Multi Resolution Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
28208	0	359	266	610	1640	0
111818	32	7140	250	609	1640	0
695072	188	104375	297	609	1750	16
1235703	250	91844	250	594	1703	0
2776953	406	215375	297	594	1687	15
11101145	765		282	609	1688	16
30831008						
69357161						

A2. Ordinateur portable Centrino

Mémoire utilisée

Mémoire utilisée Load (cells)	Image Matricielle Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
13872	88	1872	17256	15112	29204	1020
54742	420	3204	17136	15096	29000	1440
338348	2616	9012	17136	14668	28692	3508
599864	3980	25196	17172	15044	29116	9152
1349694	9960	39116	17332	15060	29116	13912
5389384	38868	157128	17296	15060	29116	55564
14960810	111224	461632	17260	16044	29116	162148
33653456	263356		17436	15128	29116	346960

Mémoire Utilisée Load (cells)	Image Multi Résolutions Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
28208	136	4296	56380	48988	81644	3064
111818	324	7832	55940	48940	81488	4040
695072	4740	36600	55644	48968	81488	10888
1235703	8804	67540	56588	48972	81488	25392
2776953	21760	146416	56196	48948	81496	43480
11101145	86908	585484	55968	48968	81488	173800
30831008	239576		56148	48968	81488	495036
69357161			56024	48972	81488	

Mémoire Utilisée Load (cells)	Image Ponctuelle Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
28208	0	5840	59156	50728	85876	3568
111818	240	13652	58500	50768	86560	6028
695072	10188	73496	59108	51420	86568	21776
1235703	19364	132876	60112	51408	86560	46028
2776953	43496	293656	59720	51408	86588	86956
11101145	172776		59480	51416	86556	347592
30831008						980736
69357161						

Overhead:	7.968665575	> 40	5.486598004	4.742752572	7.984162354	32.06283748

Temps de chargement

Loading Time — Image matricielle

Load (cells)	Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
13872	261	289	1269	6845	1051	298
54742	255	349	1238	7291	1175	297
338348	271	1200	1255	8019	2049	315
599864	270	984	1237	7005	1079	409
1349694	297	1180	1222	7201	1049	361
5389384	408	3691	1221	6899	1068	595
14960810	677	10584	1205	6832	1049	1200
33653456	1199		1205	7088	1049	2311

Loading Time — Image Multi-Résolutions

Load (cells)	Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
28208	621	835	4365	24629	2972	846
111818	623	1056	4315	24156	2991	845
695072	614	1542	4302	25809	2991	910
1235703	655	2612	4330	27266	3020	956
2776953	691	4058	4315	27169	2972	1016
11101145	945	13954	4251	25988	3035	1517
30831008	1582		4346	25769	3042	3085
69357161			4342	27333	3020	

Loading Time — Image Ponctuelle

Load (cells)	Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
28208	674	914	4549	26020	4319	911
111818	694	1399	4599	25088	3240	911
695072	694	2384	4392	25399	3242	992
1235703	741	4129	4396	25399	3256	1103
2776953	820	7398	4376	27298	3220	1299
11101145	1364		4314	24656	3238	2492
30831008						
69357161						

Temps d'exécution

Loading Time Load (cells)	Image matricielle Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
13872	0	51	173	45	73	51
54742	35	70	380	89	291	66
338348	127	176	1972	525	1636	159
599864	285	411	3709	869	2856	491
1349694	535	631	10700	1928	7186	650
5389384	2129	2433	80500	7368	26963	2661
14960810	6130	7028	41791	19999	67229	7679
33653456	14327		167943	44791	167280	22100

Loading Time Load (cells)	Image Multi Resolution Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
28208	17	151	535	151	166	145
111818	68	183	925	523	376	175
695072	381	416	6124	3082	2022	410
1235703	666	996	7539	5379	4089	1017
2776953	1521	1633	23370	11911	8657	1642
11101145	6519	6242	136709	46181	35480	8815
30831008	17948		598777	126494	96401	31657
69357161			2281011	269280	225620	

Loading Time Load (cells)	Image Multi Resolution Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
28208	65	209	935	598	380	209
111818	221	396	2475	2182	1631	393
695072	1379	1497	15501	13099	9691	1582
1235703	2506	3075	26303	22997	16938	3363
2776953	5631	6183	78101	51371	38298	6411
11101145	24305		568175	202571	152044	35094
30831008						
69357161						

Temps de déchargement

Loading Time Load (cells)	Image matricielle Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
13872	0	0	78	172	625	0
54742	0	63	94	188	610	0
338348	16	157	94	125	578	0
599864	78	361	94	141	625	0
1349694	125	516	95	125	625	0
5389384	265	1937	93	157	609	0
14960810	500	6777	94	171	610	0
33653456	796		94	171	625	32

Loading Time Load (cells)	Image Multi Resolution Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
28208	16	845	281	578	1531	0
111818	31	1234	282	578	1609	0
695072	94	84828	282	593	1563	0
1235703	140	75781	282	578	1625	0
2776953	203	170359	297	578	1609	0
11101145	453	350594	234	578	1625	16
30831008	859		282	579	1578	32
69357161			265	563	1562	

Loading Time Load (cells)	Image Multi Resolution Grid	Bucket Grid	Rtree	KD Tree	MXQuad Tree	Chained Links
28208	0	361	266	610	1640	0
111818	32	7140	251	609	1640	0
695072	188	104375	297	609	1750	16
1235703	255	91844	251	594	1703	0
2776953	409	215410	297	594	1687	15
11101145	767		283	609	1688	16
30831008						
69357161						

Bibliographie

[1] Aref, W.G. and I.F. Ilyas. *An Extensible Index for Spatial Databases.* in *International Conference on Scientific and Statistical Database Management (SSDBM'01).* 2001. Edinburgh, SCOTLAND.

[2] Bayer, R. and E. McCreight, *Organization and Maintenance of Large Ordered Indexes.* Acta Informatica, 1972. **Vol. 1**(Fasc. 3): p. pages 173-189.

[3] Beckley, D.A., M.W. Evens, and V.K. Raman, *Multikey Retrieval from K-d Trees and Quad-Trees.* ACM Digital Library, 1985.

[4] Bentley, J.L., *Multidimensional Binary Search Trees Used for Associative Searching.* Communications of the ACM, 1975. **Vol. 18**(No. 9): p. pages 509-517.

[5] Bogdanov, A., et al., *Final Report on Multi-Source Data Fusion and Algorithms.* 2005, École des Mines de Paris: Paris. p. 74 pages.

[6] Bretschneider, T. and O. Kao, *Image Fusion in Remote Sensing.* 2001, Technical University of Clausthal. p. 8 pages.

[7] David A. Fay, R.T.I., Neil Bomberger, and Allen M. Waxman. *Multisensor & Spectral Image Fusion & Mining: From Neural Systems to Applications.* in *Applied Imagery Pattern Recognition Workshop.* 2003.

[8] de Béthune, S., F. Muller, and J.-P. Donnay, *Fusion of Multispectral and Panchromatic Images by Local Mean and Variance Matching Filtering Techniques.* Fusion of Earth Data, Sophia Antipolis, 1998(January): p. pages 28-30.

[9] Fagin, R., et al., *Extendible Hashing - A Fast Access Method for Dynamic Files.* ACM Transactions on Database Systems, 1979. **Vol. 4**(No. 3): p. Pages 315-344.

[10] Fay, D.A., et al., *Fusion of Multi-Sensor Imagery for Night Vision: Color Visualization, Target Learning and Search.* 2000, Massachusetts Institute of Technology. p. 8.

[11] Finkel, R.A. and J.L. Bentley, *Quad Trees : A Data Structure for Retrieval on Composite Keys.* Acta Informatica, 1974.

[12] Foley, J., et al., *Computer Graphics: Principle and Practice.* 1990, Massachusets: Addison-Wesley.

[13] Freeston, M. *The BANG File : a new kind of grid files.* in *SIGMOD87.* 1987: ACM Press.

[14] Gao, H., *Formal Information Fusion Framework.* 2000.

[15] Gardner, S.B. and J.K. Uhlmann, *A Decentralized Data Fusion Framework for Horizontal Integration of Intelligence Data*, Naval Research Laboratory, Editor. 2005.

[16] Guttman, A., *R-Trees. A Dynamic Index Structure for Spatial Searching.* ACM Digital Library, 1984.

[17] Guttman, A., et al., *R-TREES: A DYNAMIC INDEX STRUCTURE FOR SPATIAL SEARCHING.* 2004.

[18] Hutflesz, A., H.-W. Six, and P. Widmayer, *Twin Grid Files: Space Optimizing Access Schemes.* ACM Digital Library, 1988.

[19] Khoshelham, K., *Building Extraction from Multiple Data Sources : A Data Fusion Framework for Reconstruction of Generic Models*. 2004, The Hong Kong Polytechnic University.

[20] Kiema, J.B.K. *Wavelet Compression and Data Fusion: An Investigation into the Automatic Classification of Urban Environments using Colour Photography and Laser Scanning Data*. in *International Conference on Pattern Recognition*. 2000.

[21] Kok, R.d., et al., *Object based image analysis of high resolution data in the alpine forest area*. 1999.

[22] Kriegel, H.-P. and B. Seeger, *PLOP-Hashing: A Grid File without Directory*. IEEE, 1988.

[23] Kumar, R., et al. *DFuse: A Framework for Distributed Data Fusion*. in *SenSys'03*. 2003. Los Angeles: ACM.

[24] Lindenbaum, M. and H. Samet, *A Probabilistic Analysis of Trie-Based Sorting of Large Collections of Line Segments in Spatial Databases*. 2000, Technion: Haifa, Israel. p. 32 pages.

[25] Llinas, J., *An Introduction to Multi-Sensor Data Fusion*. IEEE, 1998: p. VI-537 - VI-540.

[26] Lomet, D.B. and B. Salzberg, *The hB-tree: a multiattribute indexing method with good guaranteed performance*. ACM Transactions on Database Systems 1990. **Vol. 15**(No. 4): p. pages 625-658.

[27] M. González de Audicana, R. García, and A. Seco, *Fusion of Multispectral and Panchromatic Images using Wavelet Transform. Evaluation of Crop Classification Accuracy*., in *Departamento de Proyectos e Ingeniería Rural*. 2002, Universidad Pública de Navarra: Pamplona, Spain.

[28] Meenakshisundaram, V., *Quality Assessment of Ikonos and Quickbird Fused Images for Urban Mapping*, in *Department of Geomatics Engineering*. 2005, University of Calgary: Calgary. p. 118 pages.

[29] Nievergelt, J., H. Hinterberger, and K.C. Sevcik, *The Grid File: An Adaptable, Symmetric Multikey File Structure*. ACM Transactions on Database Systems, 1984. **Vol. 9**(No. 1): p. Pages 38-71.

[30] Office québécois de la langue française. *définition pour : classification*. 2006 [cited; Available from: http://www.granddictionnaire.com/.

[31] Ozkarahan, E.A. and M. Ouskel. *Dynamic and Order Preserving Data Partitioning for Database Machines*. in *Proceedings of VLDB 85*. 1985. Stockholm.

[32] Pohl, C. *Tools and Methods for Fusion of Images of Different Spatial Resolution*. in *International Archives of Photogrammetry and Remote Sensing*. 1999. Valladolid, Spain.

[33] Robinson, J.T. *The K-D-B-Tree: A Search Structure for Large Multidimensibnal Dynamic Indexes*. in *Proceedings of the 1981 ACM SIGMOD international conference on Management of data SIGMOD '81*. 1981: ACM Press.

[34] Sadjadi, F., *Comparative Image Fusion Analysis*. 2005, Lockheed Martin Corporation. p. 8.

[35] Samet, H. and R.E. Webber, *Storing a Collection of Polygons Using Quadtrees.* ACM Transactions on Graphics, 1985. **Vol. 4**(No. 3): p. Pages 182-222.

[36] Seeger, B. and H.-P. Kriegel. *The Buddy-Tree: An Efficient and Robust Access Method for Spatial Data Base Systems*. in *Proceedings of the 16th VLDB Conference*. 1990. Brisbane, Australia.

[37] Sheng, A., *Sensor Data Fusion Using Kalman Filters on an Evidence Grid Map*, in *School of Computer Science*. 2005, Carleton University: Ottawa. p. 185.

[38] Streilein, W., et al., *Fused Multi-Sensor Image Mining for Feature Foundation Data*. 2000, Massachusetts Institute of Technology, National Imagery and Mapping Agency. p. 8.

[39] Varshney, P.K., *Multisensor Data Fusion and Applications*, Department of Electrical Engineering and Computer Science, Editor. 2004, Syracuse University: Syracuse, NY.

[40] Vergara, O.R., *Data Fusion*. 2001, Instituto Militar de Engenharia (IME). p. 8.

[41] Wald, L. *Definitions and Terms of Reference in Data Fusion*. in *International Archives of Photogrammetry and Remote Sensing*. 1999. Valladolid, Spain.

[42] Weihe, K., *A Software Engineering Perspective on Algorithmics.* ACM Digital Library, 2001.

[43] Wikipedia. *kd-tree*. Wikipedia [cited; Available from: http://en.wikipedia.org/wiki/Kd-tree.

[44] Zhang, J., et al. *All-Nearest-Neighbors Queries in Spatial Databases.* in *International Conference on Scientific and Statistical Database Management (SSDBM'04)*. 2004. Santorini Island, Greece.