Université Nice Sophia Antipolis

POLYTECH NICE-SOPHIA

Rapport de stage
- Sciences Informatique 5e année

Titre du stage : Segmentation automatique d'images numériques issues de photographies aériennes : application à la détection et à la géo-localisation des tombes dans un cimetière
Entreprise : Équipe ICAR, Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier
Lieu du stage :

LIRMM
UMR 5506 - CC 477
161 rue Ada
34095 MONTPELLIER CEDEX 5

Étudiant : Louis Tribouillard
Maîtres de stage : Marc Chaumont, Gérard Subsol
Enseignant tuteur : Marc Antonini

Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier
LIRMM

UNIVERSITÉ MONTPELLIER 2
SCIENCES ET TECHNIQUES

Table des matières

Remerciements

Je souhaite tout d'abord remercier le LIRMM pour m'avoir permis de faire ce stage et l'équipe ICAR pour son accueil.

Plus particulièrement je remercie mes deux tuteurs M. Marc Chaumont et M. Gérard Subsol pour leurs disponibilités, leurs conseils avisés, leurs réponses à toutes mes questions ainsi que pour leur suivi durant le stage.

Je tiens aussi à remercier M. Laurent Deruelle, qui était responsable de ce projet au sein de Berger-Levrault pour sa présence, son aide et ses retours sur mon travail.

Je remercie aussi mon tuteur enseignant M. Marc Antonini pour ses retours concernant le rapport et pour m'avoir suivi tout au long du stage.

Enfin je remercie les administrations du LIRMM et de Polytech'Nice pour leur écoute et l'aide qu'ils m'ont apportée.

Chapitre 1

Introduction

Ce stage se déroule au sein de l'équipe ICAR du laboratoire de recherche du LIRMM [1] et s'inscrit dans le cadre de la collaboration de recherche UM2 n°122129 entre la société Berger-Levrault et le LIRMM.
L'objectif de cette collaboration est d'étudier la segmentation automatique d'images numériques obtenues par photographie aérienne pour la détection et la géo-localisation des tombes dans un cimetière.
En effet l'entreprise Berger-Levrault propose de réaliser des cadastres précis des cimetières identifiant toutes les tombes. Ce travail est réalisable depuis le sol par un géomètre, mais ce travail n'est pas facile ni exhaustif à cause de l'imprécision du GPS. Ainsi, il est plus aisé d'utiliser des images aériennes pour réaliser ces cadastres.

Figure 1.1 – Exemple d'image aérienne utilisée au cours du stage.

Cependant, pour étudier une image aérienne d'un cimetière afin de délimiter les tombes, il faut

1. Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier

relever manuellement la position de celles-ci sur l'image, ce qui peut se révéler très fastidieux (un cimetière peut facilement posséder plus de 1000 tombes).
Pour son logiciel de gestion concernant les administrations publiques locales [2], la société Berger-Levraut souhaiterait concevoir un programme capable d'automatiser la segmentation des tombes (dans un premier temps, puis de divers autres objets) afin de gagner du temps dans la réalisation de cette tâche.
Ce stage est la suite d'un stage de l'année dernière, réalisé par Florian Courtade (étudiant à Polytech'Nice-Sophia).
L'année dernière, Florian Courtade avait étudié deux méthodes visant à détecter et segmenter les tombes d'une image aérienne. Il avait étudié une méthode basique, utilisant les lignes de partage des eaux et une autre méthode d'apprentissage fondée sur les travaux de Viola & Jones[8]. La seconde méthode, Viola & Jones, est deux fois plus performante que la première méthode (respectivement une précision, un rappel et un f-score de 0.23, 0.24 et 0.24 pour la méthode basée sur les ligne des partage des eaux et 0.72, 0.49 et 0.53 pour la seconde méthode).
Cependant, pour passer à une automatisation, un grand nombre d'améliorations doivent être proposées. C'est dans ce but que le LIRMM a proposé ce stage. Après une étude de l'état de l'art réalisée par Marc Chaumont durant 3 mois (qui a débouché sur le document *Dossier préliminaire du contrat UM2 n°122129*), un étudiant (moi-même) devait étudier une autre méthode plus pratique et théoriquement plus performante pour détecter les tombes d'un cimetière. Ainsi, le but de mon stage est d'étudier la méthode issue de la publication d'Aldavert et al. : *Fast and Robust Object Segmentation with the Integral Linear Classifier* [3][3] afin de la comparer aux deux méthodes du stage précédent.

Dans ce document, je présenterai le travail qui m'a été proposé au début du stage, puis je passerai sur le travail réalisé avant de conclure.

2. `http://www.berger-levrault.fr/solutions/fonction-publique-territoriale/services-citoyens/e-cimetiere.html`

3. Méthode générale expliquée sur `http://www.cvc.uab.cat/~aldavert/plor/`

Chapitre 2

Travail proposé

2.1 Introduction

Ce stage a pour but la compréhension, l'utilisation et l'amélioration de la méthode de segmentation proposé par Aldavert et al.[3] afin de la comparer aux travaux réalisés l'année dernière par Florian Courtade (Lignes de partage des eaux et Viola & Jones).
Cet article est paru en 2009 et présente une méthode de segmentation générique d'objets fondée sur un apprentissage de même performance que l'état de l'art de l'époque avec une complexité calculatoire moindre.

Le document *Dossier préliminaire du contrat UM2 n°122129* rédigé par Marc Chaumont au cours des 3 mois précédant mon stage présente un échantillon de l'état de l'art, et présente donc des méthodes qui pourraient être utiles au projet et qui apportent de nouvelles idées.
Il faut savoir que la recherche d'objets particuliers appliquée à des tombes n'a jamais été réalisée et il n'existe pas d'article traitant directement de ce problème. Les méthodes repérées sont donc des méthodes de segmentation utilisées pour détecter d'autres types d'objet.
Par exemple, l'article de [12] propose de localiser des places de parking à partir d'une image aérienne. L'approche consiste à segmenter des images en régions via l'algorithme mean-shift (après être passé dans l'espace de couleur cie-luv), puis extraire des informations de régions et utiliser un SVM linéaire pour réaliser l'apprentissage et la classification.
Le second article intéressant est celui de [9], qui propose de détecter des rectangles en utilisant les transformées de Hough (cf partie 3.4.3).
On peut aussi citer l'article de Harzallah, Jurie et Schmid[7], qui propose d'utiliser un SVM linéaire dans un premier temps puis d'affiner par un SVM non linéaire (descripteurs utilisés : HOG (cf 3.1.1)).
Enfin la méthode d'Aldavert[3] est celle qui a été retenue, les résultats affichés semblent être à la hauteur de l'état de l'art et les codes sources sont en lignes.

La méthode d'Aldavert et al. est une méthode utilisée pour détecter des objets dans une image. Dans son article, Aldavert l'utilise pour détecter des vélos, des voitures ou encore des personnes (images standards dans la base GRAZ02[1]).
Nous allons essayer, une fois la méthode comprise et le code fonctionnel, d'utiliser cette méthode pour la reconnaissance de tombes depuis une image aérienne.

2.2 Compréhension

Avant de pouvoir utiliser le programme d'Aldavert (le code source était disponible sur le site), il s'agissait de comprendre les différentes étapes des algorithmes utilisés. L'étude de l'article d'Aldavert, des articles annexes (concernant des algorithmes repris par Aldavert, par exemple les ERF[5]) ainsi que du code source devait occuper les premières semaines.
Les explications relatives aux algorithmes de la méthode sont présentées dans la partie 3.1.

2.3 Utilisation

La méthode comprise, il fallait installer le logiciel fourni (ainsi que toutes les librairies nécessaires) puis le tester afin de voir si les résultats obtenus étaient satisfaisants. Nous verrons dans la partie 3.2 que la première version du logiciel n'était pas fonctionnelle et qu'il a fallu attendre une seconde version avant d'obtenir les premiers résultats. De plus le second logiciel devait fournir une segmentation objet (c'est à dire qu'une tombe devait être délimité par un polygone) mais ne nous fournissait qu'une segmentation pixel (chaque pixel est associé à une probabilité d'être une tombe). Nous verrons dans la partie 3.2 que le passage de la segmentation pixel à la segmentation objet (qui n'est pas implémenté dans le logiciel d'Aldavert) n'est pas facile et que la solution proposée par Aldavert (utilisation d'un Mean-Shift) n'est pas convaincante dans notre cas, le nombre de tombes ainsi que la faible distance les séparant rend l'obtention d'une segmentation objet assez délicate.

2.4 Comparaison

Une fois les premiers résultats obtenus, il s'agissait de les comparer aux résultats du stage précédent afin de se positionner par rapport aux valeurs que Florian Courtade avait obtenues. La prise en main des logiciels développés par Florian Courtade ainsi que l'installation des librairies requises étaient donc nécessaires pour effectuer la comparaison. Comme nous l'avons vu, les résultats donnés par le programme d'Aldavert étaient sous la forme de carte probabiliste représentant la probabilité pour chaque pixel d'être une tombe. Afin d'effectuer une comparaison qui ait du sens, il était nécessaire de transformer cette carte de probabilité en segmentation objet.

2.5 Amélioration

Les comparaisons effectuées, il était prévu d'améliorer le programme d'Aldavert avec, par exemple, l'ajout de nouveaux descripteurs, etc. Cependant, le logiciel n'effectuant pas de segmentation objet, il a fallu travailler plus en détail sur ce point afin d'obtenir des résultats utilisables. En effet, un premier passage de la carte de probabilité à une segmentation objet a été réalisé afin d'obtenir une première comparaison avec les programmes de Florian Courtade assez rapidement, mais ce passage était très simpliste et peu performant. La phase d'amélioration a donc été utilisé pour développer une méthode plus efficace pour l'obtention de la segmentation objet.

Chapitre 3

Travail réalisé

Le déroulement du stage s'est déroulé pratiquement comme le planning (cf section 3.6) le prévoyait sauf la partie " amélioration ".

3.1 Compréhension

La première partie du stage concernait l'étude des divers documents que mes tuteurs de stages (Marc Chaumont et Gérard Subsol) avaient mis à ma disposition afin de me faciliter la compréhension globale du stage. Ainsi, j'ai commencé par lire les dossiers préliminaires concernant les contrats liant le LIRMM et l'entreprise Berger-Levrault. Ces documents résultent d'une étude de 3 mois précédant chaque stage (celui de Florian Courtade ainsi que le mien) et présentent l'état de l'art et les différentes publications qui pourraient être utiles concernant le problème de segmentation des tombes à partir d'images aériennes. Le planning des stages est aussi défini dans ces documents.
La lecture de ces documents m'a permis de bien saisir le sujet ainsi que le domaine dans lequel j'allais évoluer.
Par la suite, j'ai commencé l'étude de la méthode d'Aldavert par le biais de sa publication *Fast and Robust Object Segmentation with the Integral Linear Classifier*[3]. Cette publication explique brièvement les différents algorithmes utilisés et fournit des résultats obtenus sur les bases d'images GRAZ02[1] et VOC2007[10].
Le site d'Aldavert [1] reprend la publication et fournit le code source correspondant à sa méthode.
Parallèlement à la lecture de la publication, l'étude du code source ainsi que de publications annexes (*Randomized clustering forests for image classification*[5], *Extremely Randomized Trees*[13]) m'a permis de comprendre en détail la méthode avant de l'utiliser.

Les étapes de la méthode d'Aldavert

La méthode utilisée par Aldavert peut se découper comme ceci :
- L'acquisition des caractéristiques de l'image : pour ce faire, Aldavert a essayé 3 descripteurs différents : le descripteur SURF[6], le descripteur IHOG[14] et le descripteur Integral Shape Context[15].
  Le descripteur IHOG étant le plus performant d'après la comparaison réalisée par Aldavert, c'est celui que j'ai étudié.
- La génération d'un dictionnaire afin d'utiliser le principe des sacs de mots (transformation des descripteurs extraits en mots visuels). Aldavert a implémenté deux méthodes permettant la génération de ce dictionnaire : la méthode Hierarchical K-Means[11] et la méthode Extremely Randomized Forest[5].

1. `http://www.cvc.uab.cat/~aldavert/plor/`

La méthode Extremely Randomized Forest étant la plus performante d'après l'article, c'est celle-ci que j'ai étudiée.
- L'utilisation d'un classifieur pour déterminer la classe à partir des mots visuels.
- L'utilisation d'un algorithme type Mean-shift (algorithme utilisé pour repérer les régions homogènes : on passe d'une classification par pixel la segmentation d'un objet)

3.1.1 Acquisition des caractéristiques de l'image

Afin de traiter une image, il faut d'abord d'extraire des informations de celle-ci. Il existe plusieurs types de descripteurs pour une image (texture, couleur, etc...). Dans notre cas, on va étudier les descripteurs IHOG (Integral Histogram Of Gradients).
Ce sont des descripteurs HOG (Histogram Of Gradient) sur lesquels on utilise le principe d'image intégrale pour augmenter la rapidité de calcul.

Descripteur HOG

Ces descripteurs sont en fait des vecteurs de 32 valeurs scalaires qui sont extraits de région de l'image (cellules carrés de X x X pixels, en bleu sur la figure ci dessous). Ces 32 valeurs correspondent à 8 valeurs de gradient pour 8 orientations (0° 45° ... 315°) par zones (il y a 4 zones par cellule).
Ils sont extraits de manière dense (c'est à dire qu'on va travailler sur toute l'image, à l'inverse d'une méthode dite parcimonieuse où l'on sélectionnerait des zones d'intérêts), on les obtient en parcourant l'image avec notre cellule de X x X pixels en utilisant un pas inférieur à la taille de la cellule, afin d'être plus précis.
Dans l'image suivante, on voit apparaître les cellules en bleu, et chaque cellule est découpée en 4 parties desquelles seront extraites les informations d'orientation (module du gradient selon l'orientation).

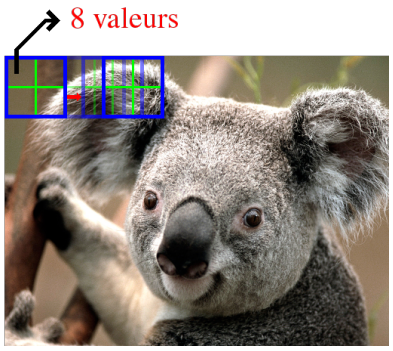


Figure 3.1 – La grille est en bleu et on prend les informations d'orientation pour chaque sous-zone

Pour obtenir le module de chaque gradient ainsi que sa dérivée directionnelle, on va parcourir les cases de la grille à l'aide d'ondelettes de Haar (sous la forme d'éléments plus petits encore que la cellule) qui vont nous donner les gradients en X et en Y.

Figure 3.2 – Ondelette de Haar en x et en y

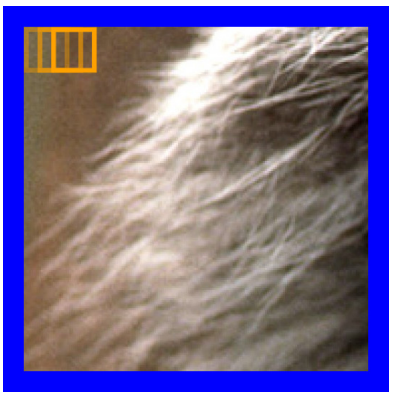


Figure 3.3 – On parcourt chaque cellule pour obtenir les gradients

On peut déduire de ces 2 mesures le module du gradient ainsi que la dérivée directionnelle. On projette le module du gradient sur les deux orientations les plus proches (parmi les 8 orientations du descripteur : 0° 45° ... 315°). On va ensuite stocker ces deux informations dans 2 des 8 tableaux contenant le module des gradients selon leurs orientations. Une fois que la cellule a été complètement parcourue et que les tableaux contenant les orientations sont complets, on va pouvoir extraire le descripteur HOG en sommant puis normalisant les modules des gradients pour chaque orientation pour chaque sous-zone. Dans la figure suivante on peut voir les tableaux contenant les modules des gradients, ceux ci sont découpés en 4 pour l'extraction du descripteur HOG (les modules sont sommés puis normalisés).

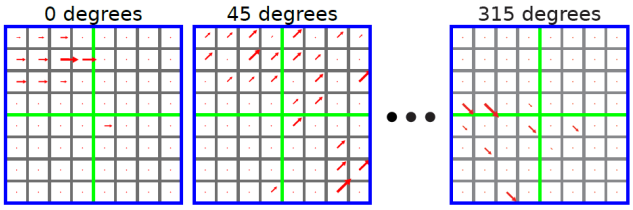


Figure 3.4 – Il y a 8 tableaux de modules de gradients pour chaque cellule de la grille (en bleu)

Integral HOG

Tous ces calculs pour l'extraction des descripteurs sont exécutés de nombreuse fois (environ 2000 fois par image, pour une image 640*480).
Afin d'accélérer le processus, le principe d'image intégrale est utilisé.

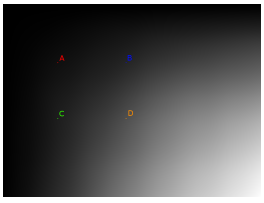


Figure 3.5 – Exemple d'image intégrale ; Les 4 points ABDC délimitent une zone rectangulaire.

Une image intégrale c'est une représentation sous la forme d'une image, de même taille que l'image d'origine, qui en chacun de ses points contient la somme des pixels situés au dessus et à gauche de ce point. Ce qui entraîne que n'importe quelle somme de pixel d'une région de l'image intégrale peut être calculée avec seulement 4 additions.

Comme on peut le voir dans la figure suivante, on peut obtenir la somme des pixels d'un rectangle à partir de seulement 4 points de l'image intégrale.

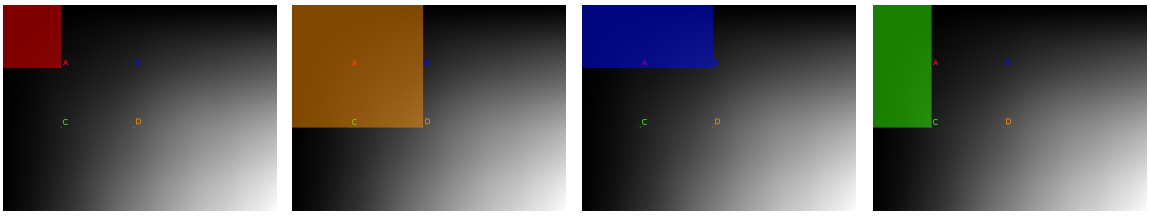


FIGURE 3.6 – L'expression $D + A - B - C$ nous donne la somme des pixels présents dans le rectangle ABDC.

Ce principe est utilisé d'une part au niveau des pixels de l'image, afin de faciliter l'obtention des gradients à l'aide des ondelettes de Haar, mais aussi au niveau des orientations de gradient. En effet, les images contenant les orientations sont en fait des images intégrales, ce qui permet de calculer plus rapidement la somme des gradients dans les zones.

3.1.2 Génération du dictionnaire

Le nombre de descripteurs obtenus à l'étape précédente est assez important (de l'ordre de 70000 descripteurs pour une image 640*480) et cela peut poser problème pour l'entraînement du classifieur. Aldavert propose donc une méthode afin de quantifier ces descripteurs. Pour cela il utilise une technique appelée ERF (Extremely Randomized Forest) qui va lui permettre d'obtenir des mots visuels à partir des descripteurs. Cette méthode se base sur des arbres créés à partir de descripteurs dont on connaît la classe (tombe / pas tombe). Ces arbres seront utilisés pour la quantification des descripteurs. Ainsi un descripteur va traverser chaque arbre de la racine jusqu'aux feuilles et une unique feuille par arbre sera associée au descripteur. Cela permettra de quantifier tous les descripteurs en mots visuels.

Construction d'un arbre :
Soit un ensemble de N descripteurs (vecteurs de 32 valeurs caractéristiques) dont on connaît la classe. Un vecteur contient 32 valeurs scalaires : $\mathbf{v} = (c_1, c_2, .., c_{32})$

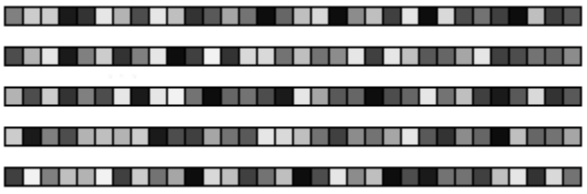


FIGURE 3.7 – 5 descripteurs

On imagine que l'on connaît les classes des vecteurs, par exemple les vecteurs v_1, v_3, v_5 appartiennent à une classe tandis que les deux autres n'y appartiennent pas.
On va choisir au hasard la i^e valeur : c_i

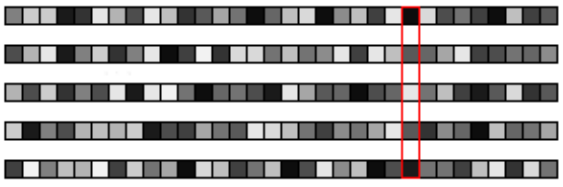


FIGURE 3.8 – On prend la i^e valeur

On effectue pour chaque c_i un test booléen T : $\{c_i > \sigma\}$ avec σ un scalaire choisi au hasard, par exemple 20.

FIGURE 3.9 – Valeur correspondant a 20

Comme on peut le voir dans la figure ci dessous, l'ensemble $\{v_1, v_2, v_3, v_4, v_5\}$ est donc partitionné en deux sous-ensembles $\{v_1, v_5\}$ et $\{v_2, v_3, v_4\}$ selon le test booléen T.

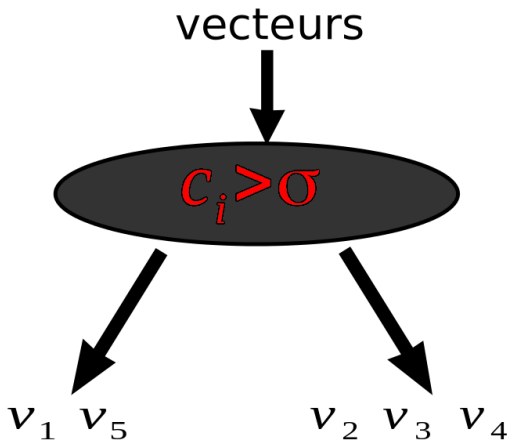


FIGURE 3.10 – On obtient deux sous-ensembles

Soit L la partition $\{\{v_1, v_5\}, \{v_2, v_3, v_4\}\}$, on va vérifier si celle ci est convenable en utilisant une formule basée sur l'entropie de Shannon [13] :

$$Sc(L,T) = \frac{2 \cdot I_{C,T}(L)}{H_C(L) + H_T(L)} \tag{3.1}$$

Où :

H_C représente l'entropie caractérisant la répartition des classes dans L :

$$H_C(L) = -\sum_{c \in C} \frac{n_c}{n} \log_2 \frac{n_c}{n} \tag{3.2}$$

n est la taille de l'ensemble L, n_c est le nombre de descripteurs de L appartenant à la classe C

H_T représente l'entropie caractérisant le test booléen T :

$$H_T(L) = -\sum_{p=1}^{2} \frac{n_p}{n} \log_2 \frac{n_p}{n} \tag{3.3}$$

n_p est le nombre de descripteurs de L appartenant à l'un ou l'autre des ensembles séparés par le test booléen

$I_{C,T}$ représente l'information mutuelle de la séparation :

$$I_{C,T}(L) = H_C(L) - \sum_{p=1}^{2} \frac{n_p}{n} H_C(L_p) \tag{3.4}$$

La procédure de séparation d'un ensemble s'effectue en boucle (c'est à dire que lorsque la partition ne valide pas le test d'entropie, on annule cette partition et on en recrée une autre en prenant une autre caractéristique et un autre test booléen au hasard) jusqu'à ce que la partition obtenue possède un score supérieur à un seuil S_{min} ou jusqu'à ce qu'un nombre d'itération T_{max} soit atteint
Si la partition est convenable, on applique l'algorithme sur les deux sous-ensembles obtenus. Un ensemble correspond à une feuille lorsque tous les vecteurs présents dans celui ci appartiennent à la même classe ou lorsqu'un seul sous-ensemble est obtenu, c'est à dire que toutes les caractéristiques c_k des vecteurs de l'ensemble valident le test booléen T.
Dans notre cas, par exemple, le premier sous-ensemble obtenu $\{v_1, v_5\}$ contient deux vecteurs de la même classe, on obtient une première feuille. Pour l'autre sous-ensemble $\{v_2, v_3, v_4\}$, on peut appliquer l'algorithme une fois de plus :
On choisit au hasard une valeur : la j^e

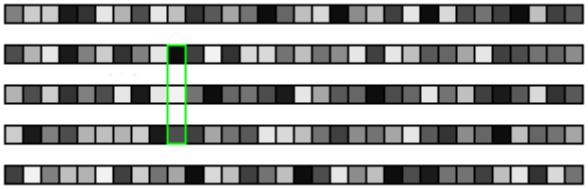
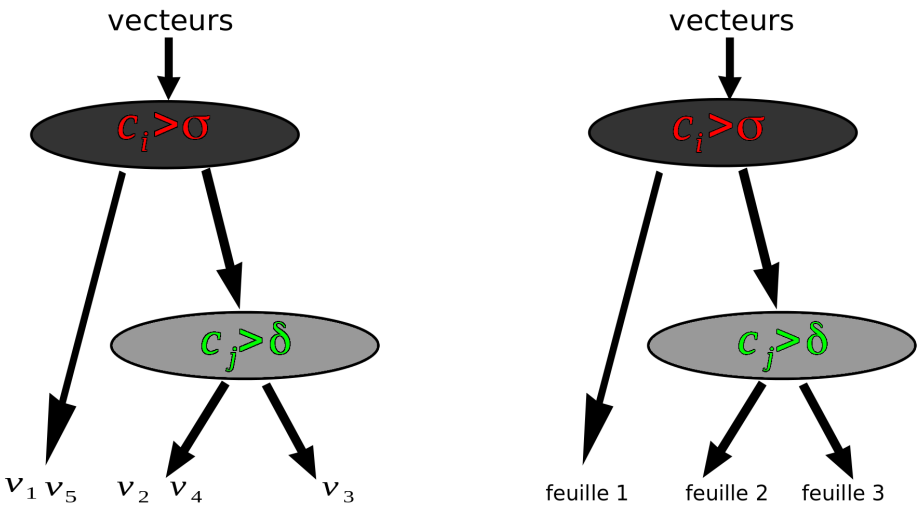


Figure 3.11 – On prend la j^e valeur des 3 vecteurs de l'ensemble

On choisit un seuil δ au hasard, par exemple 60

Figure 3.12 – Couleur correspondante à 60

Comme on le voit dans la figure ci dessous, en comparant la j^e caractéristique c_j de chaque vecteur de notre sous-ensemble $\{v2, v3, v4\}$ à δ, on obtient une nouvelle partition.



(a) Partitionnement final des 5 vecteurs

(b) Arbre final

Figure 3.13 – La construction de l'arbre est terminée, on a maintenant un outil pour quantifier nos vecteurs.

Les deux nouveaux sous-ensembles $\{v2, v4\}$ et $\{v3\}$ sont convenables, on peut donc arrêter. On a obtenu notre arbre qui va permettre la quantification des vecteurs caractéristiques.
L'arbre va associer un vecteur a une feuille en se basant sur les tests de chaque nœud.

Construction de la forêt :

La forêt comprend N arbres obtenus avec cet algorithme (l'algorithme étant en majeure partie créé aléatoirement, les arbres ne seront pas les mêmes). L'utilisation de plusieurs arbres permet une quantification plus robuste.

Quantification :

Une fois la forêt obtenue, on peut quantifier un descripteur en lui faisant parcourir tous les arbres de la racine jusqu'aux feuilles. Un descripteur va donc déboucher sur N mots visuels (chacun correspondant à une feuille de la forêt, une feuille par arbre). Pour chaque pixel, on va sélectionner tous les descripteurs présents dans son voisinage (zone carrée de XxX pixels définie par l'utilisateur) et on va obtenir pour chaque descripteur les N mots visuels qui lui correspondent. A l'aide de ces mots visuel on va pouvoir définir un histogramme représentant la fréquence d'apparition de chaque mot.
Un pixel est donc représenté par une distribution (histogramme) de mots visuels.
Cet histogramme sera ensuite envoyé au classifieur pour l'entraînement et/ou la classification.

3.1.3 Le classifieur

Une fois les histogrammes obtenus, on peut entraîner un classifieur afin de pouvoir prédire l'appartenance d'une région à telle ou telle classe.
Pour ce faire, Aldavert utilise un classifieur basique : un LR-SVM [2] qu'on entraînera avec un échantillon annoté d'images aériennes. La phase de classification, une fois l'apprentissage effectué, comprend quant à elle une petite astuce : encore une fois l'utilisation du principe d'image intégrale.
Normalement, pour l'obtention du score d'un pixel, nous devrions calculer l'histogramme des mots visuels présents dans le voisinage de ce pixel (on prendra tous les mots visuels correspondant aux pixels présents dans la zone rectangulaire définie par ce voisinage) puis envoyer cet histogramme au classifieur.
Mais ici, nous allons commencer par calculer le score pour chaque pixel (c'est à dire qu'on va prendre l'ensemble des mots visuels qui correspondent à ce pixel) puis, pour calculer le score d'une région, nous utiliserons le principe de l'image intégrale pour obtenir le score de la région plus facilement.
En effet, les régions pour lesquelles est calculé le score se superposent lorsque l'on passe d'un pixel à son voisin, on a donc une redondance des mots visuels situés dans le voisinage de deux pixels adjacents qu'il est possible d'ignorer en utilisant le principe de l'image intégrale.

Pour calculer le score permettant de classer une région, il faut effectuer le produit scalaire entre l'histogramme de la région $\vec{H}$ et le vecteur des poids du LR-SVM $\vec{W}$ (en ajoutant un biais b). On obtient donc ce type de formule :

$$score = \frac{1}{\left\|\vec{H}\right\|} \sum_{i=0}^{n} h_i w_i + b \tag{3.5}$$

Où h_i est la i^e composante du vecteur $\vec{H}$ et w_i est la i^e composante du vecteur $\vec{W}$.
On peut appliquer le principe de l'image intégrale seulement si tous les w_i sont positifs, on définit donc :

$$\tilde{w_i} = w_i - W_{min} \tag{3.6}$$

2. Logistic Regression Support Vector Machine

(2.5) devient donc :

$$score = \frac{1}{\left\|\vec{H}\right\|} \sum_{i=0}^{n} h_i \tilde{w}_i + \frac{W_{min}}{\left\|\vec{H}\right\|} \sum_{i=0}^{n} h_i + b \tag{3.7}$$

Si on utilise la norme 1 : $\frac{1}{\left\|\vec{H}\right\|} \sum_{i=0}^{n} h_i = 1$, on peut donc simplifier (3.7) :

$$score = \frac{1}{\left\|\vec{H}\right\|} \sum_{i=0}^{n} h_i \tilde{w}_i + W_{min} + b \tag{3.8}$$

On peut maintenant utiliser le principe de l'image intégrale pour calculer $\sum_{i=0}^{n} h_i \tilde{w}_i$ plus rapidement.
En effet soit $L_c(x, y)$ la somme des poids $\tilde{w}_i$ avec l'histogramme des mots visuels correspondant au pixel (x,y) et L_c l'image représentant toutes ces sommes a chaque pixel. On peut créer une image intégrale de ces sommes (que l'on nommera I_c).
On peut maintenant calculer le résultat du classifieur pour n'importe quelle zone rectangulaire de l'image avec simplement 4 additions. En effet le score pour une région R sera :

$$score = \frac{1}{\left\|\vec{H}\right\|} H_R + W_{min} + b \tag{3.9}$$

avec

$$H_R = I_c(x_0, y_0) + I_c(x_1, y_1) - I_c(x_1, y_0) - I_c(x_0, y_1) \tag{3.10}$$

Où $I_c(x_0, y_0)$ et $I_c(x_1, y_1)$ sont respectivement les coordonnées du coin supérieur gauche et du coin inférieur droit de la région R.

En utilisant ce principe, on optimise la vitesse de la classification de tous les pixels d'une image.

3.2 Utilisation

Une fois la méthode comprise, je me suis lancé dans l'installation du code source d'Aldavert ainsi que des divers librairies requises (installation de Linux, OpenCV, modification légère du makefile, etc). Après l'installation, j'ai dû constituer une base de données à partir des images de cimetières que m'avaient fournies mes tuteurs (images fournies par Berger-Levrault pour le stage de l'année précédente).
Les images constituant la base de donnée proviennent toutes de cimetières de petits villages de la Haute-Marne, 52 (plus précisément de villages situés dans les environs de Langres).
J'ai souhaité suivre la méthodologie de l'article et j'ai donc voulu garder la résolution des images de la base GRAZ02[1]. Par conséquent, j'ai découpé les images provenant d'images aériennes d'assez grande taille (en moyenne 4000*4000 pixels) en images plus petites de 640*480 pixels à l'aide d'un programme que j'ai développé en Java. Ensuite, manuellement, j'ai sélectionné des images contenant des tombes, et d'autres sans tombes, afin d'entraîner le classifieur.

Ces images de cimetières disposaient déjà de la vérité terrain associée (c'est à dire un document, image ou texte, qui contient les positions réelles des tombes) sous forme d'image où les pixels blancs correspondent à une tombe (comme on peut le voir dans l'image ci dessous). J'ai donc découpé ces images de la même manière que les images aériennes pour finalement obtenir une base d'images de 640*480 pixels avec les vérités terrain associées.

La base de données utilisée pour obtenir les résultats présentés dans ce rapport contient 100 images (d'une résolution de 640*480 pixels), dont 60 contenant effectivement des tombes.
L'apprentissage se fait sur toute l'image, ainsi même sur une image contenant des tombes, des descripteurs caractérisant l'arrière plan ou des objets différents d'une tombe sont calculés et traduits en mots

FIGURE 3.14 – Exemple d'une image de la base de données avec la vérité terrain associée.

visuels qui seront envoyés au classifieur.
Une seconde base de données, appelée base de validation (contenant 50 images, dont 30 de tombes) est utilisée pour régler automatiquement les paramètres du classifieur.

Après la constitution de cette base de données, j'ai lancé les premiers apprentissages puis j'ai testé le programme. Malheureusement, celui ci ne fonctionnait pas pour la reconnaissance pixel. J'ai essayé pendant quelques temps de trouver le problème, mais n'y arrivant pas, j'ai préféré prendre contact avec Aldavert pour savoir si celui ci avait des réponses à mes problèmes. Il m'a très vite répondu que le code présent sur son site était obsolète et qu'il pourrait me fournir sous peu une version plus récente de son programme.

Après la réception du nouveau code, j'ai installé les programmes nécessaires à la compilation du code ainsi que les nouvelles librairies requises (toc [3] , zfstream [4] , etc) et après une petite modification du makefile, le programme était fonctionnel. Cette fois ci le programme a donné les premiers résultats, sous forme de carte de probabilité.
Bien sur, les tests sont effectués sur des images de cimetières n'appartenant pas à la base de données d'apprentissage.

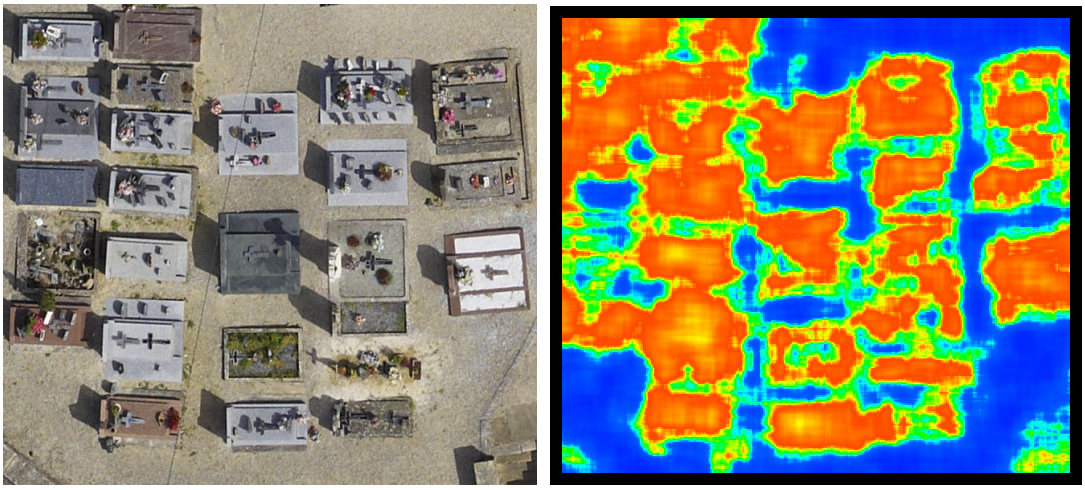


FIGURE 3.15 – Partie du cimetière de Lecey ainsi que la carte de probabilité associée.

La carte de probabilité nous donne une segmentation pixel, il me fallait une segmentation objet. Avant de pouvoir effectuer les premières comparaisons j'ai donc du trouver un moyen de passer les cartes de probabilité en segmentation objet.

3. `http://toc.sourceforge.net/`
4. `http://wanderinghorse.net/computing/zfstream/`

On peut voir sur l'image précédente que les zones correspondants aux tombes (en orange/rouge) ne sont pas forcément bien séparées. Aldavert préconisait l'utilisation d'un Mean-Shift pour la segmentation objet. J'ai donc programmé un algorithme mean-shift pour vérifier les résultats mais ceux ci n'était pas satisfaisant : la carte de probabilité était trop approximative pour obtenir quelque chose de correct. En effet nous obtenions des groupes de tombes qui étaient reliés pour n'en former qu'une seule. Un simple filtrage selon la probabilité nous donnait quasiment la même chose.
A cette époque, mes tuteurs ont souhaité participer au congrès Digital Heritage 2013 [5] et ont commencé à rédiger un article à soumettre. Pour terminer cet article pour lequel j'assistais mes tuteurs, il nous fallait absolument une segmentation objet. En prenant le filtrage selon la probabilité et en prenant le rectangle englobant de chaque zone obtenue, j'ai rapidement obtenu une première segmentation objet.

Le traitement réalisé sur la carte de probabilité à été réalisé en Java car c'est un langage que je maîtrise et qui permet de réaliser très vite un programme fonctionnel.

Cette méthode permettait de passer de la carte de probabilité à une segmentation objet.

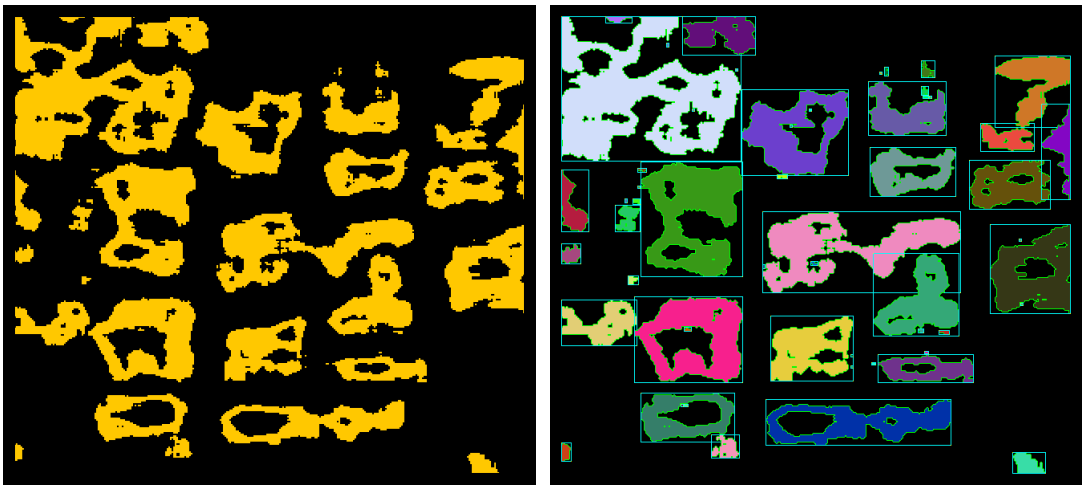


Figure 3.16 – Filtrage selon la probabilité puis acquisition des différentes zones.

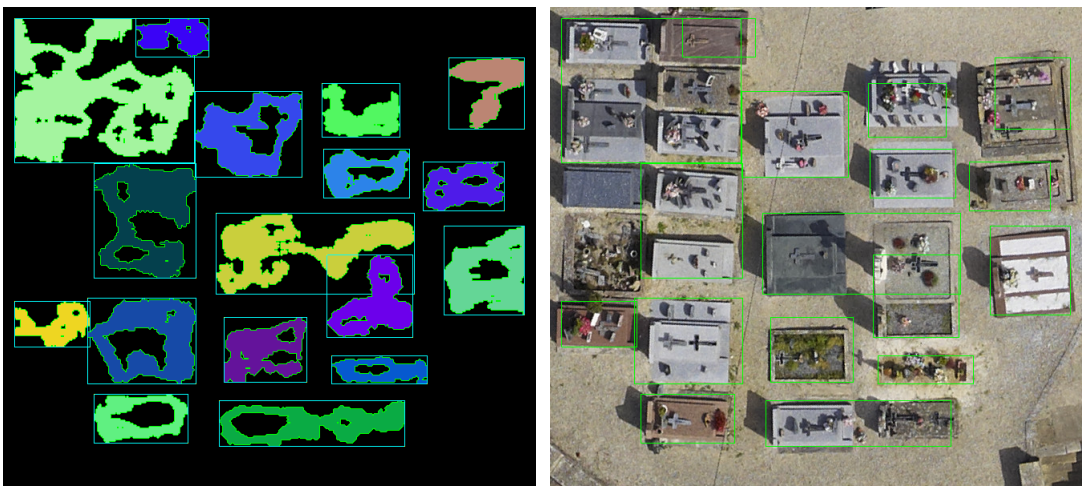


Figure 3.17 – Suppression des zones trop petites et première segmentation objet.

Mais la segmentation objet obtenue était très approximative et segmentait souvent plusieurs tombes ensemble. J'ai donc décidé de filtrer chaque rectangle englobant pour détecter plus précisément les tombes à l'intérieur.

5. `www.digitalheritage2013.org`

Figure 3.18 – Exemple d'une zone, et des contours détectés.

Ainsi pour chaque zone, j'ai effectué un filtrage à l'aide d'ondelettes de Haar pour obtenir les contours des tombes.
Une fois les contours obtenus, j'ai isolé les centres des rectangles (en fixant une limite basse pour la taille des rectangles à détecter) puis j'ai pu tracer un contour plus précis pour chaque tombe.

J'ai donc obtenu une nouvelle segmentation objet, un peu plus précise que la précédente, et suffisante pour l'article et pour effectuer les premières comparaisons, cependant, le passage de la segmentation pixel à la segmentation objet était encore loin d'être optimal (en effet dans ce cas précis, l'algorithme ne détectait que les tombes verticales ou horizontales).

Figure 3.19 – Segmentation objet finale.

Avec cette première segmentation, j'ai pu d'une part donner des premiers résultats pour l'article et d'autre part j'ai pu me pencher sur la partie comparaison.

3.3 Comparaison

Pour effectuer une comparaison qui ait du sens, j'ai du me familiariser avec les logiciels de Florian Courtade afin de pouvoir tester la même image avec toutes les méthodes.
Florian Courtade ayant déjà développé un programme calculant le rappel, la précision et le F-score d'une segmentation (cf annexe A), j'ai décidé de le réutiliser non seulement pour ses propres méthodes, mais aussi pour la méthode d'Aldavert. Ainsi j'ai programmé un petit logiciel qui traduit les résultats d'une segmentation via la méthode d'Aldavert dans un format txt et j'ai modifié un peu le code de Florian (C++) pour qu'il puisse lire le fichier .txt et utiliser les données ainsi obtenues pour calculer précision, rappel et F-score de la même manière que pour ses méthodes. Pour effectuer j'ai commencé par prendre le cimetière de Signy (cimetière ne figurant ni dans la base d'apprentissage de la méthode d'Aldavert ni dans la base de la méthode type Viola & Jones de Florian Courtade). Cette image nous a été fournie cette année par Berger-Levrault. Elle ne comprenait donc pas la vérité terrain qui est nécessaire pour effectuer le calcul de la précision, du rappel et du F-score.
Afin de simplifier l'obtention de cette vérité terrain (qui dans notre cas correspond à relever les coordonnées des différents sommets du polygone correspondant à la tombe ; les images fournies comprenant

parfois des centaines de tombes, la tâche peut être fastidieuse), j'ai réalisé un petit logiciel en Java permettant de simplifier la relève de ces points (interface graphique représentant l'image du cimetière et qui relève automatiquement les points sélectionnés).

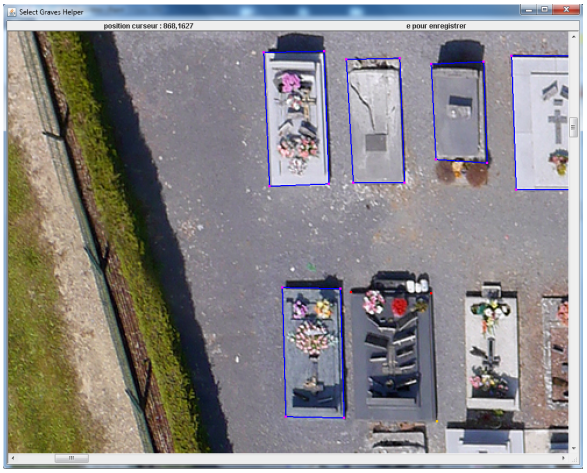


FIGURE 3.20 – Logiciel d'aide pour la création de la vérité terrain.

Une fois la vérité terrain obtenue, j'ai pu effectuer les premières comparaisons :

FIGURE 3.21 – Segmentation obtenue avec la méthode d'Aldavert.

FIGURE 3.22 – Segmentation obtenue avec la méthode Viola & Jones.

Comme on peut l'observer sur les images précédentes, la détection basée sur Viola & Jones est très performante sur les tombes verticales (même si elle n'est pas parfaite). Cependant celle-ci est vite limitée lorsque les tombes ne sont plus verticales (cf FIGURE 3.23). En effet, pour son logiciel, Florian Courtade s'était entraîné sur une base de donnée contenant uniquement des tombes verticales (l'entraînement de plusieurs classifieurs selon différentes orientations était prévu mais l'apprentissage prenant environ 2 semaines de calcul, cela ne s'est pas fait).

FIGURE 3.23 – A gauche, détection via la méthode d'Aldavert. A droite, détection via la méthode de Viola & Jones.

Concernant les résultats pour l'ensemble du cimetière, on obtient pour la méthode Viola & Jones :
- précision : 0.72449
- rappel : 0.581967
- f-score : 0.605802

Pour ce qui est des résultats concernant la méthode d'Aldavert :
- précision : 0.763912
- rappel : 0.530445
- f-score : 0.564979

Dans les deux cas, les résultats ne sont pas satisfaisant pour une automatisation. Cependant la méthode d'Aldavert reste correcte qu'importe l'orientation de la tombe ce qui est avantageux. Mais d'un autre coté, pour des tombes verticales, elle est en moyenne bien moins efficace que la méthode type Viola & Jones qui détecte très bien les tombes verticales.
Ainsi on passe à la partie suivante : l'amélioration.

3.4 Amélioration

A la base l'amélioration devait porter sur l'étude et l'ajout de nouveaux descripteurs, etc. Cependant, le passage de la segmentation pixel à la segmentation objet étant très basique et peu efficace, c'est sur cette partie que j'ai travaillé.
Afin de réaliser une segmentation objet plus précise, j'ai décidé de m'attaquer au post-traitement appliqué à la carte de probabilité. Le nombre de tombes ainsi que le faible espace qui les sépare rend l'utilisation de la carte de probabilité délicate, en effet, comme nous pouvons le voir, il n'est pas rare que les tombes soit collées sur la carte de probabilité. Ce qui amène à un effet de regroupement de tombes lorsque l'on effectue le premier seuillage. Il s'agit alors de déterminer dans chaque rectangle englobant où sont les tombes.

FIGURE 3.24 – Partie du cimetière de Signy.

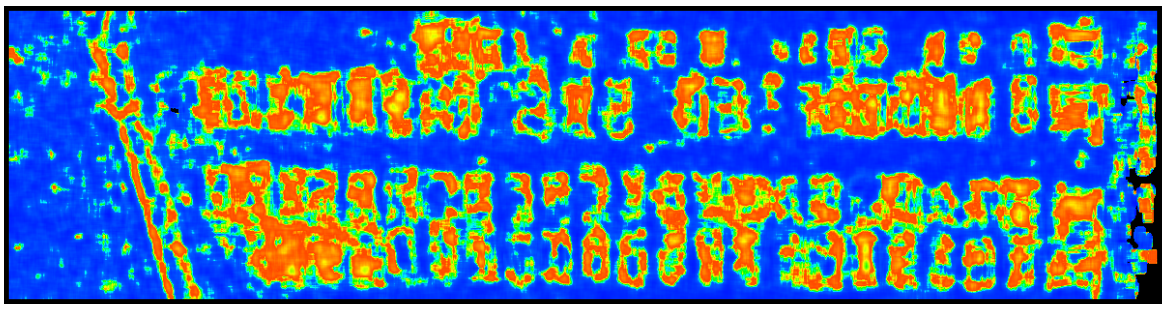


FIGURE 3.25 – Carte de probabilité associée.

Dans ce but, j'ai décidé d'utiliser un filtre de Canny[2] ainsi que d'utiliser un système similaire à celui présenté dans l'article *Rectangle Detection based on a Windowed Hough Transform*[9] sur l'image filtrée, à savoir : la détection de rectangle (une tombe est souvent rectangulaire) à l'aide de la transformée de Hough.

3.4.1 Canny

La première étape consiste en l'obtention d'un filtrage binaire plus précis que le filtrage utilisé jusqu'à maintenant (pour rappel : utilisation des ondelettes de Haar en X et Y pour obtenir les contours). Dans ce but, nous utiliserons donc le filtrage de Canny. Cette technique, développée en 1986, est un filtre très connu qui permet la détection de contours.

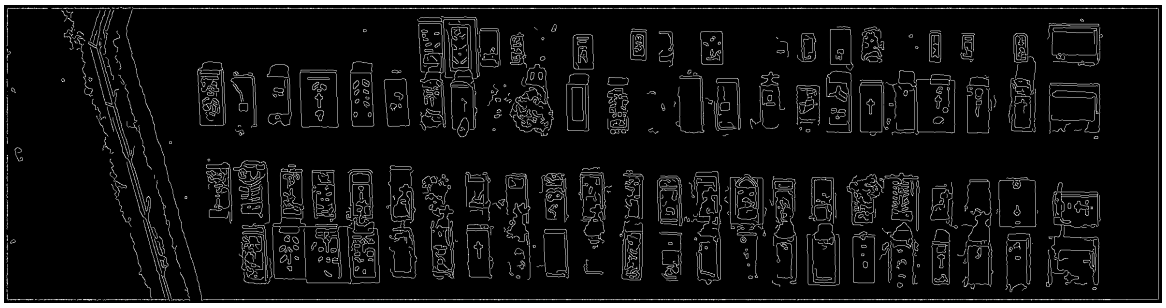


FIGURE 3.26 – Filtre de Canny appliqué sur l'image précédente.

Le filtre de Canny[2] fonctionne grâce à l'utilisation d'un filtrage (type Sobel) afin d'obtenir une estimation du gradient de l'image puis d'un seuillage par hystérésis : c'est à dire que l'on va définir 2 seuils que l'on va comparer à l'intensité du gradient. Si le gradient d'un point est supérieur au seuil haut, on va le marquer en tant que contour, si il est inférieur au seuil bas, on va le rejeter (il ne fera jamais parti du contour) et si il est entre les deux seuils, on va le considérer comme un élément de contour seulement si il est juxtaposé à un point considéré comme étant un contour.

Grâce à ce filtre, nous obtenons (comme nous pouvons le voir dans la figure plus haut) une image binaire représentant les contours des tombes.

Afin d'appliquer le filtre de Canny, il faut fixer les seuils haut et bas (ainsi que régler le flou gaussien que l'on applique préalablement, mais ceci à une importance moindre). Par exemple, si l'on réduit les deux seuils haut et bas, on peut obtenir un filtrage moins lisible (FIGURE 3.27).

Cependant, après un filtrage qui retire tous les segments trop petits pour être des contours de tombes (paramètre que l'on peut fixer), on peut remarquer que les images issues de n'importe quelle filtrage de Canny (sauf ceux avec des seuils vraiment trop haut) seront à peu près similaires (FIGURE 3.29).

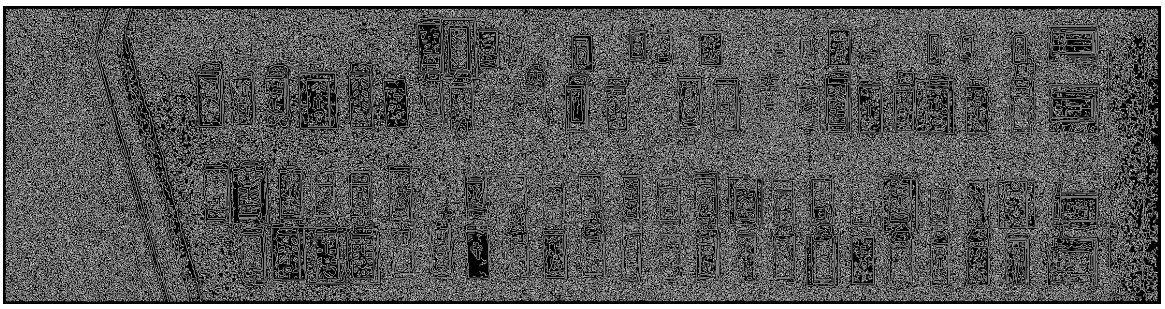


FIGURE 3.27 – Filtre de Canny avec des seuils peu élevés.

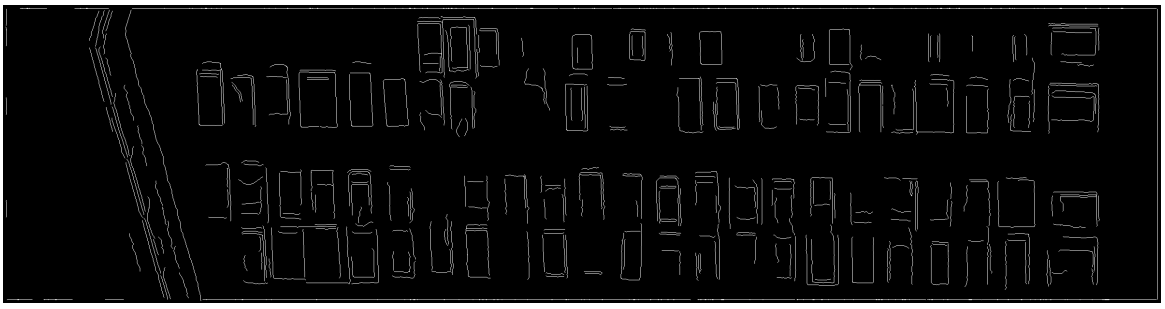


FIGURE 3.28 – Les segments de petite taille de la FIGURE 3.26 ont été retirés.

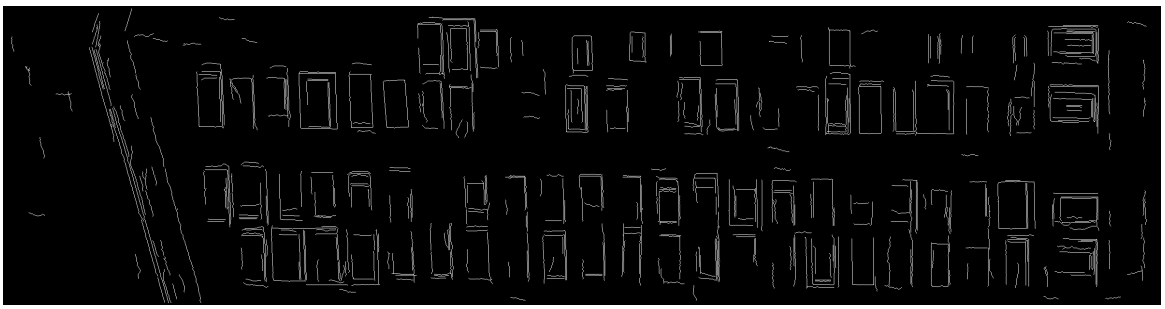


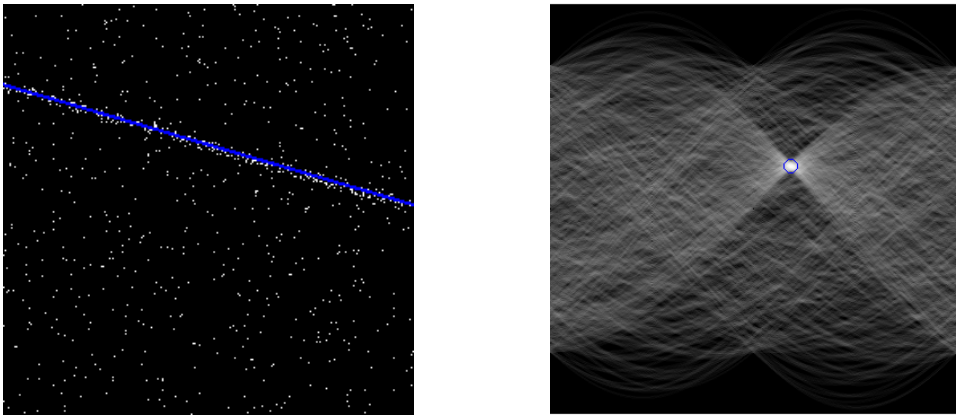
FIGURE 3.29 – Les segments de petite taille de la FIGURE 3.27 ont été retirés.

On va pouvoir utiliser les transformées de Hough afin de détecter les lignes principales et possiblement détecter les rectangles que forment les tombes.

3.4.2 Transformée de Hough

La transformée de Hough est une technique qui a été inventée en 1962 par Paul Hough. La transformée de Hough telle que nous l'utilisons à été inventée par R. Duda et P. Hart[4]. A la base cette technique permet de reconnaître les lignes dans une image.
Pour ce faire, on va représenter chaque point de l'image dans l'espace de Hough.
A chaque point est associé un nombre infini de lignes qui passent par ce point (seul l'angle θ est différent).
En calculant la valeur ρ (qui est la distance de l'origine à la ligne) pour chaque θ, on va obtenir une sinusoïde dans l'espace polaire (ρ en fonction de θ).
Chaque point de ce nouvel espace (espace de Hough) représente une droite dans l'espace cartésien.
Si l'on effectue cette transformation pour chaque point d'une image, on va obtenir un ensemble de sinusoïdes dans l'espace de Hough (qui représenteront l'ensemble des droites passant par chaque point de l'image initiale).
L'intersection de ces sinusoïdes indiquera une droite formée par les points dans l'espace cartésien.

En pratique, l'angle θ et la distance ρ sont des notions discrètes, on peut ainsi obtenir un tableau $C(\rho_k, \theta_l)$ qui contiendra le nombre de points présents sur chaque droite (représentée par le couple ρ, θ).



(a) Droite détectée en bleu.

(b) Intersection correspondant à la droite.

FIGURE 3.30 – A gauche, image initiale ; A droite, espace de Hough correspondant.

Les maximums locaux de ce tableau représentent les intersections des sinusoïdes dans l'espace de Hough et représente donc les droites sur l'image initiale.

3.4.3 Détection des rectangles

Afin de détecter les rectangles dans l'image, nous allons utiliser la méthode proposée par Claudio Rosito Jung and Rodrigo Schramm dans leur publication *Rectangle Detection based on a Windowed Hough Transform*[9].
Dans ce but, nous allons utiliser les caractéristiques des rectangles dans l'espace de Hough.

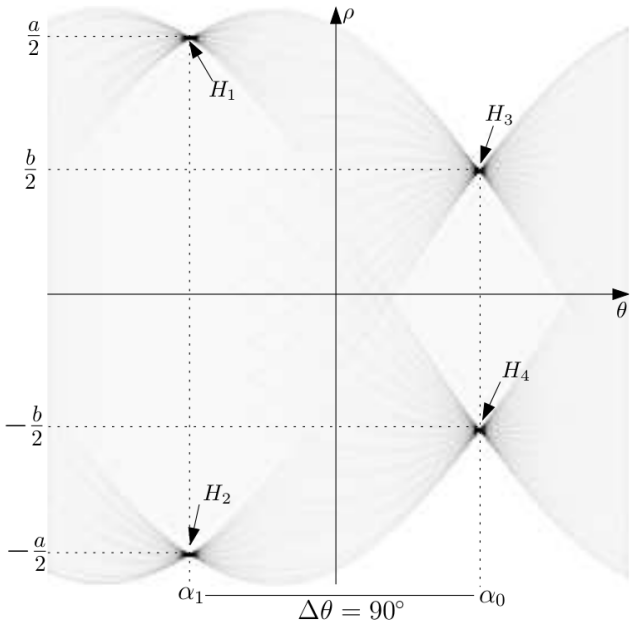


FIGURE 3.31 – Espace de Hough correspondant à une image contenant un rectangle.

Prenons par exemple 4 intersections dans l'espace de Hough (chacune correspondant à l'une des droites passant par un coté d'un rectangle de longueur a, et de largeur b).
Si on note ces intersections $H1, ..., H4$ avec $H1 = (\rho_1, \theta_1), ..., H4 = (\rho_4, \theta_4)$
Ces 4 intersections satisfont certaines relations géométriques :

- Les intersections apparaissent par paire, chacune située à un θ particulier (α_1 pour $H1$ et $H2$ et α_0 pour $H3$ et $H4$).

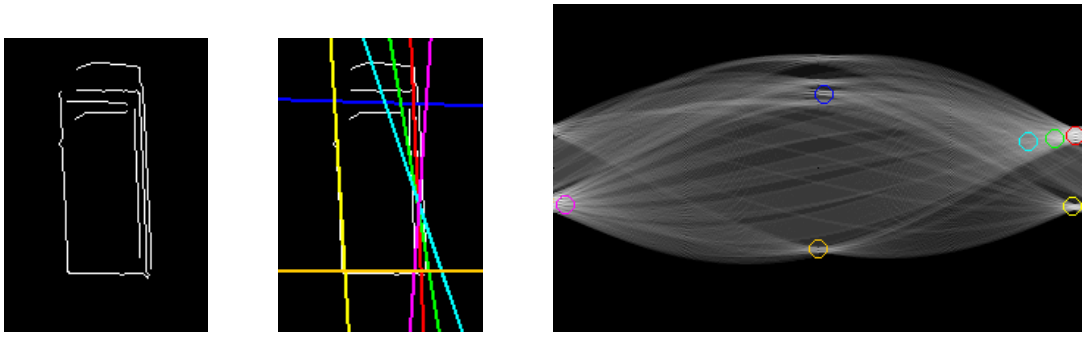
– Les deux paires sont distantes de 90° sur l'axe des θ ($\Delta\theta = 90°$).

– Deux intersections appartenant à la même paire sont symétriques par rapport à l'axe des θ (c'est à dire que $\rho_1 + \rho 2 = 0$ et $\rho_3 + \rho 4 = 0$).

– La distance (ρ) entre deux intersections d'une même paire représente la longueur d'un des cotés du rectangle ($\rho_1 - \rho_2 = a$ et $\rho_3 - \rho_4 = b$)

– Les valeurs du tableau C correspondent aux longueurs des côtés du rectangle : $C(\rho_1, \theta_1) = C(\rho_2, \theta 2) = b$ et $C(\rho_3, \theta_3) = C(\rho_4, \theta 4) = a$ (Cette relation n'est réellement exacte que si les rectangles sont entièrement tracés et qu'il n'y pas de bruit sur l'image).

On notera que ces relations ne sont valables que si le rectangle se trouve (à peu près) au centre de l'image étudiée (en particulier la relation n°3).



(a) Image initiale. (b) Droites détectées. (c) Intersections correspondant aux droites.

FIGURE 3.32 – Tombe obtenue avec le filtre de Canny, et l'espace de Hough correspondant.

Pour tester l'existence d'un rectangle dans la fenêtre que l'on étudie, il faut commencer par scanner chaque intersection H_i.
Deux intersections H_i et H_j seront regroupées en paire si elles satisfont les conditions suivantes :
– $\Delta\theta = |\theta_i - \theta_j| < T_\theta$
– $\Delta\rho = |\rho_i + \rho_j| < T_\rho$
– $|C(\rho_i, \theta_i) - C(\rho_j, \theta_j) < T_L \frac{C(\rho_i,\theta_i)+C(\rho_j,\theta_j)}{2}$

La première condition vérifie que les intersections H_i et H_j sont des lignes parallèles ($\theta_i \approx \theta_j$), plus le seuil T_θ est petit, plus les lignes devront être parallèles pour qu'elles soient validées.
La seconde condition vérifie que les lignes correspondant à H_i et H_j sont symétriques par rapport à l'axe des θ ($\rho_i \approx -\rho_j$).
Enfin la dernière condition vérifie que les deux lignes ont approximativement la même longueur ($C(\rho_i, \theta_i) \approx C(\rho_j, \theta_j)$).

Une fois que toute les paires de l'image sont constituées, on va comparer deux à deux la moyenne de leurs angles, $\alpha_k = \frac{1}{2}(\theta_i + \theta_j)$, afin de trouver les paires qui sont orthogonales.
Ainsi un rectangle est détecté si deux paires dont la moyenne des angles vaut α_k et α_l respectent la condition $\Delta\alpha = ||\alpha_k - \alpha_l| - 90| < T_\alpha$.

Une fois qu'un rectangle est détecté, on peut déterminer les points grâce aux équations de droite de chaque intersection :

$$y = \frac{-cos(\theta_1)}{sin(\theta_1)}x + \frac{\rho_1}{sin(\theta_1)} \tag{3.11}$$

$$y = \frac{-cos(\theta_2)}{sin(\theta_2)}x + \frac{\rho_2}{sin(\theta_2)} \tag{3.12}$$

Il est possible d'isoler x, puis de trouver y.

$$x = \frac{\frac{\rho_2}{sin(\theta_2)} - \frac{\rho_1}{sin(\theta_1)}}{\frac{-cos(\theta_1)}{sin(\theta_1)} - \frac{-cos(\theta_2)}{sin(\theta_2)}} \tag{3.13}$$

On obtient ainsi les 4 points de notre rectangle.

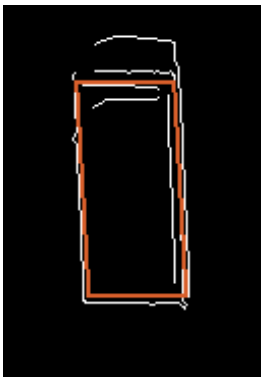


FIGURE 3.33 – Rectangle déterminé à partir des droites trouvées précédemment.

Ainsi, pour une image donnée, si le rectangle se trouve au centre de celle ci, il est possible, en suivant cette méthode de trouver une segmentation objet.
Cependant, un cimetière ne se résume pas à une seule tombe, et celle ci n'est pas forcément au milieu de l'image. Il est donc nécessaire de parcourir l'image avec une région dans laquelle on cherchera l'existence d'un rectangle.
Cette région doit être assez grande pour contenir n'importe quel rectangle, mais aussi la plus petite possible pour éviter les côtés des tombes adjacentes.
Une région convenable serait un cercle de diamètre externe D_{max}. Ce diamètre serait égal à la plus grande diagonale de rectangle que l'on souhaite détecter.
On peut aussi rajouter un diamètre interne D_{min} pour retirer du bruit potentiel qui se situerait à l'intérieur du rectangle. Ce diamètre étant égal à la plus petite diagonale de rectangle que l'on souhaite détecter.
La région optimale serait donc un disque de diamètre externe D_{max} et de diamètre interne D_{min}.
Il s'agit alors de parcourir l'image avec cette région pour détecter tous les rectangles possibles.

Par exemple pour l'image suivante :

FIGURE 3.34 – Sous région d'un cimetière contenant plusieurs tombes.

On va commencer par effectuer le filtrage :

Figure 3.35 – Sous région d'un cimetière contenant plusieurs tombes filtrée par Canny.

Puis on va parcourir l'image avec notre disque et on va rechercher un rectangle à chaque nouvelle région :

Figure 3.36 – Résultat de la recherche de rectangle.

On remarque que le parcours de l'image avec un pas assez faible va résulter en une démultiplication du nombre de rectangles trouvés (on va trouver plusieurs fois le même). Il s'agit alors de supprimer les doublons.
Pour cela on va calculer une mesure d'erreur (en fonction des $\Delta\rho, \Delta\theta, \Delta\alpha$ calculés précédemment), et on choisira le rectangle qui minimise cette erreur dans une certaine région (Figure 3.37).

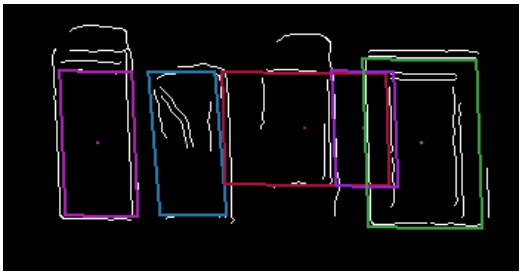


Figure 3.37 – Résultat après suppression des doublons.

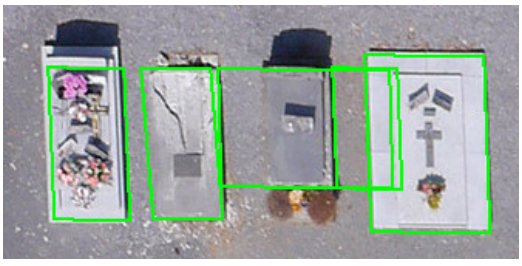


Figure 3.38 – Résultat après suppression des doublons sur l'image initiale.

On observe déjà qu'avec ce cas peu compliqué, le résultat n'est pas parfait et on a deux faux positifs. Lorsque que l'on a une sous région contenant plus de tombes, le résultat sera encore moins exploitable (Figure 3.39).

FIGURE 3.39 – Résultat après suppression des doublons sur l'image initiale.

3.5 Bilan

3.5.1 Conclusion sur les divers méthodes

A la base, le programme d'Aldavert devait fournir une segmentation objet et non une segmentation pixel. Cependant le programme que j'ai récupéré ne permettait pas de faire ceci. J'ai donc essayé de trouver un moyen d'obtenir cette segmentation objet.
Si la carte de probabilité avait été facilement lisible, la tâche aurait été plus aisée (par exemple sur des images avec une plus haute résolution). Mais ce n'était pas le cas, nous avons donc cherché une méthode permettant d'extraire une segmentation objet des ensembles de tombes que l'on obtient avec la carte de probabilité.
Nous avons vu par exemple la détection de contours simples ou la détection de rectangle via la transformée de Hough.
La seconde méthode semblait vraiment prometteuse mais au final, la proximité des tombes pose non seulement problème pour la méthode d'Aldavert, mais aussi pour des méthodes sans apprentissage comme cette méthode fonctionnant avec les transformées de Hough.
Il est prévu de faire une comparaison entre la méthode d'Aldavert combinée à la méthode utilisant la détection de rectangle et les résultats de Florian Courtade obtenus l'année passée. Cependant, au vu des résultats obtenus, on se doute que la méthode type Viola & Jones sera probablement plus performante.

Ainsi, aucune des méthodes étudiées dans ce rapport ne représente une véritable avancée par rapport à l'algorithme de Viola & Jones utilisé l'année précédente (si ce n'est en terme de temps d'apprentissage : 8h contre 2 semaines). Mais toutes ces méthodes ne sont pas forcément à jeter, en effet, elles peuvent encore être améliorées et adaptées pour être plus performantes dans le cas de cimetière.

3.5.2 Problèmes rencontrés

La partie portant sur l'étude théorique de la méthode d'Aldavert n'a pas posé de problèmes particuliers si ce n'est quelques difficultés sur la compréhension mais qui furent de courte durée.
La phase d'utilisation du premier programme d'Aldavert à été un moment difficile, en effet, le programme ne fonctionnait pas et il fallait étudier avec précision le code source pour espérer détecter une éventuelle erreur. Le code ($>$ 20000 lignes), bien que commenté, n'était pas forcément facile d'accès et le problème pouvant être n'importe où, j'ai passé plusieurs jours sans trouver de solution avant de me résoudre à contacter Aldavert lui même.
Seule la réception de la nouvelle version du code m'a sorti de ce problème. Le nouveau code était en effet un petit peu plus difficile à compiler (installation de librairies très peu connues et peu documentées) mais il était fonctionnel.

La seconde grande difficulté de ce stage est le passage de la segmentation pixel à la segmentation objet. En voyant la carte de probabilité et en visualisant plutôt bien les différentes tombes, on pourrait se dire que c'est un travail aisé. Mais ce n'est pas le cas.
En effet, l'obtention d'une bonne segmentation objet (dans les limites de ce que nous donne la carte de probabilité, si une tombe n'est pas détectée avec la segmentation pixel, elle ne le sera pas non plus lors du passage à la segmentation objet) est difficile à réaliser. Après une première méthode d'obtention assez simpliste et peu performante, il a fallu étudier des méthodes plus difficiles à mettre en place.

3.5.3 Amélioration possible

La méthode d'Aldavert, bien que prometteuse, n'a pas permis une amélioration convenable par rapport à la méthode type Viola & Jones. En effet, bien que la méthode d'Aldavert ne soit pas sensible à l'orientation des tombes, la carte de probabilité obtenue ne permet pas une segmentation du même niveau de précision que la méthode de Florian Courtade.

Le principal problème de la méthode d'Aldavert est la taille minimale des descripteurs et des zones utilisées pour obtenir les histogrammes. Cela pose problème sur la plupart des images parce que les tombes sont très rapprochées, ce qui empêche le programme de bien classer les pixels présents entre les tombes. On se retrouve donc avec une carte de probabilité peu précise (formant des "groupes" de tombes).
Sur des images d'une plus grosse résolution, je pense que la méthode d'Aldavert serait bien plus précise et segmenterait effectivement chaque tombe.
Une autre solution à envisager serait de modifier le code d'Aldavert pour réduire la taille minimum des zones utilisées pour les descripteurs et pour l'obtention des histogrammes (mais l'espace entre certaines tombes étant vraiment faible, cela entraînerait une diminution trop importante des descripteurs).

On peut aussi imaginer une méthode hybride entre Aldavert, Hough et Viola & Jones, on utiliserait la méthode d'Aldavert pour détecter les régions composées de tombes, avec les transformées de Hough, on pourrait détecter l'orientation globale des tombes (en effet, en général les tombes proches sont de la même orientation) puis positionner celles ci de manière à ce qu'elles soient verticales et enfin lancer la reconnaissance objet avec la méthode de Viola & Jones.

3.6 Planning

Voici le planning tel qu'il était prévu :

Le planning de l'étude se décompose en deux parties.
L'objectif de la première partie est de se familiariser à l'environnement, à la méthodologie d'analyse des performances sur une grande base de données, et au code d'Aldavert et al.[3]. Ce premier travail devrait prendre environ 2 mois. Voici les points successifs à aborder :

- *Lecture de la documentation préliminaire (2 semaines) :*
  - *le dossier préliminaire du contrat UM2 n°110918 (janvier-mars 2012, écrit par Marc Chaumont),*
  - *le dossier de Florian Courtade du contrat UM2 n°110918 (avril-septembre 2012) ainsi que les " slides " de présentation,*
  - *le dossier préliminaire du contrat UM2 n°122129 (janvier-mars 2013)*
  - *l'article de Aldavert et al.[3] Cette partie prendra un peu plus de temps (1 semaine) mais c'est une première lecture.*
- *Prise en main des logiciels développés par Florian Courtade (1 à 2 semaines) : installation de la librairie openCV, prise en main des exécutables :*
  - *l'exécutable de la ligne de partage des eaux,*

  - *l'exécutable de Viola & Jones,*
  - *l'exécutable d'évaluation des performances.*
- *Le code de Aldavert et al.[3] (1 mois)*

La deuxième partie consistera à mieux comprendre la méthodologie d'Aldavert et lever les ambiguïtés de la méthode. Ce travail occupera le temps associé au reste du stage, c'est-à-dire environ 4 mois. Voici les points successifs qui seront abordés :

- *Lancer des expérimentations (de la même façon que ce qui a été effectué lors du stage de Florian Courtade) dont l'objectif sera de comparer les trois méthodes : Ligne de partage des eaux, Viola et Jones, et Aldavert et al.. Les divers paramétrages de la méthode de Aldavert seront évalués.*
- *Comprendre plus finement les différentes étapes de la méthode et rédiger des explications plus détaillées que l'article initial.*
- *Proposer de nouvelles caractéristiques et les tester. Les publications récentes seront utilisées pour proposer ces caractéristiques. Quelques lectures bibliographiques additionnelles seront donc nécessaires.*

Le planning a globalement été respecté. L'étude et l'installation du code d'Aldavert a pris un peu plus de temps que prévu étant donné qu'il a fallu attendre une seconde version du code pour avoir quelque chose de fonctionnel.
La fin du stage, en particulier la partie " amélioration " diverge un peu. En effet, nous devions étudier de nouvelles caractéristiques mais nous avons choisi de nous focaliser sur la partie segmentation objet (qui n'était pas implémentée dans le programme d'Aldavert).

Chapitre 4

Conclusion

Ainsi se conclut mon stage au sein du LIRMM et dans le cadre de la collaboration avec Berger-Levrault. Ce stage était très satisfaisant, il possédait un sujet atypique et m'a fait approfondir mes connaissances sur le domaine de l'imagerie. De plus j'ai pu découvrir le monde de la recherche et le travail dans un laboratoire.

Pendant 6 mois j'ai découvert de nouvelles méthodes (notamment en segmentation d'images mais pas seulement), j'ai approfondi certaines méthodes vues en cours, et j'ai accumulé plus d'expérience dans la création et l'étude de logiciels (C++, Java).

Ce stage m'a également permis de faire des rencontres, que ce soit des chercheurs (l'équipe ICAR en particulier, ou Aldavert), ou des ingénieurs de chez Berger-Levrault (Laurent Deruelle).

Bien que les méthodes étudiées ne se soient pas montrées performantes malgré leur potentiel théorique, celles ci nous montrent que les méthodes sans apprentissage sont globalement moins robustes (par exemple, le filtre de Canny et l'utilisation des transformées de Hough), elles ne sont aussi pas toutes à jeter (la méthode d'Aldavert serait bien plus performante sur une image à plus haute résolution) et elles pourront peut être donner des pistes pour la thèse qui va suivre ce stage, toujours dans le cadre de la collaboration entre le LIRMM et Berger-Levrault.

Bibliographie

[1] M. Fussenegger A. Opelt, A. Pinz and P. Auer. Generic object recognition with boosting. *PAMI*, pages 416–431, 2006.

[2] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8 :679–714, 1986.

[3] Ricardo Toledo David Aldavert, Arnau Ramisa and Ramon Lopez De Mantaras. Fast and robust object segmentation with the integral linear classifier. In *23rd IEEE Conference on Computer Vision & Pattern Recognition (CVPR '10)*, pages 1046–1053, San Fransisco, Etats-unis, 2010. IEEE Computer Society.

[4] R. Duda and P. Hart. Use of the hough transform to detect lines and curves in pictures. *Communications of the ACM*, pages 11–15, January 1972.

[5] E. Nowak F. Moosmann and F. Jurie. Randomized clustering forests for image classification. *PAMI*, 30(9) :1632–1646, 2008.

[6] T. Tuytelaars H. Bay, A. Ess and L. van Grool. Surf : Speeded up robust features. *CVIU*, 110 :346–359, March 2008.

[7] Frédéric Jurie Hedi Harzallah and Cordelia Schmid. Combining efficient object localization and image classification. In *International Conference on Computer Vision*, sept 2009.

[8] M. Jones and P. Viola. Fast multi-view face detection. In *Technical Report 096*, Mitsubishi Electric Research Laboratories, 2003.

[9] Claudio Rosito Jung and Rodrigo Schramm. Rectangle detection based on a windowed hough transform. *Proceedings of the Computer Graphic and Image Processing*, 110 :113–120, 3 2004.

[10] C. K. I. Williams J. Winn M. Everingham, L. Van Gool and A. Zisserman. The pascal visual object classes challenge 2007 (voc2007) results. http ://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[11] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, pages 2161–2168. IEEE Computer Society, 2006.

[12] M. Ozan-Kabak O. Turgut. Parking spot detection from aerial images. *Tech. Rep., Stanford*, 2010.

[13] D. Ernst P. Geurts and L. Wehenkel. Extremely randomized trees. *Machine Learning J.*, 63, 2006.

[14] K. Cheng Q. Zhu, M. Yeh and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *CVPR*, pages 1491–1498. IEEE Computer Society, 2006.

[15] J. Malik S. Belongie and J. Puzicha. Shape matching and object recognition using shape context. *PAMI*, 24 :509–522, 4 2002.

Annexe A

Formules de comparaison

Voici les formules de comparaison telles que présentées dans le rapport de stage de Florian Courtade :

Pour quantifier les résultats, nous allons utiliser plusieurs indicateurs :

- Vrais positifs (VP) : quand l'algorithme nous renvoie une tombe qui en est bien une sur la réalité terrain.
- Faux positifs (FP) : quand l'algorithme nous renvoie une tombe qui n'en est pas une sur la réalité terrain.
- Rappel : Le rappel est défini par le nombre de régions pertinentes retrouvées au regard du nombre de régions pertinentes que possède la base de données. C'est une mesure de sensibilité.

$$Rappel = \frac{VP}{\Omega}$$

avec Ω le nombre de tombes de la réalité terrain

- Précision : La précision est le nombre d'objets pertinents retrouvés rapporté au nombre d'objets total retourné par la détection. Plus la précision est élevée, plus le résultat sera fiable.

$$Precision = \frac{VP}{VP + FP}$$

- F-score : Le f-score combine la précision et le rappel en permettant de favoriser l'un ou l'autre grâce à un critère β. Ici on prendra 2 pour favoriser le rappel sur la précision. On préfère en effet un algorithme qui nous renvoie quelques faux positifs plutôt qu'un qui « oublierait » des tombes.

$$F - score = \frac{(1+\beta)^2 \cdot (Precision \cdot Rappel)}{\beta^2 \cdot Precision + Rappel}$$

Abstract

This document is the report for my end-of-course internship. I worked at the LIRMM (Montpellier, France) for the ICAR team and the Berger-Levrault company. The goals of this six month internship were to study a segmentation method developped by Aldavert et al.[3], to implement this method in order to automatically segment graves from aerial images of cemeteries, to improve this method, and finally to compare my results with those from a previous internship which took place a year ago and see if an automation was possible.

In this document, we will see a complete explanation of the Aldavert's method and the results we got. We will also explain different methods to get an object segmentation from the pixel segmentation resulting from Aldavert's method.

To conclude, even if the results were not as good as we expected, I learnt a lot from this internship and, hopefully, my work will be a basis for a future thesis under the supervision of the ICAR team and the Berger-Levrault company.