



GéoPeuple

<i>Rapport numéro</i>	L3.1 - L3.3-1
<i>Titre</i>	Spécifications des images de Cassini pour la reconnaissance automatique de symboles et premiers résultats
<i>Rédigé par</i>	Jonathan Guyomard (LIP6)
<i>État (final / en cours)</i>	Final
<i>Relu par</i>	Matthieu Cord / Nicolas Thome / Thierry Artières
<i>Date</i>	Mars 2012

Table des matières

1	Contexte	3
2	Introduction	4
3	Présentation générale de la méthodologie	4
3.1	Spécifications des images	4
3.2	Fenêtre glissante	5
3.3	Descripteurs d'images	5
3.3.1	Distance de Chanfrein - Contours	5
	Algorithme de Canny pour le calcul des contours	5
3.3.2	Histogrammes de gradients orientés	6
	Normalisation Gamma/couleur	6
	Calcul des gradients	7
	Calcul des histogrammes	7
3.3.3	Descripteur contextuel	8
	Calcul du descripteur	9
	Analyse qualitative/quantitative	9
	Cas concret	9
3.4	Apprentissage supervisé	9
3.4.1	Outil d'apprentissage : Support Vector Machin (SVM)	9
3.4.2	Gestion des hard-négatifs	10
4	Évaluation des performances	11
4.1	Mesures de la précision et du rappel	12
4.2	Précision moyenne (Average precision)	12
5	Expérimentations et résultats	12
5.1	Création d'une interface graphique	12
5.2	Intérêt des hard-négatifs	12
5.3	Détection des objets ponctuels	13
5.3.1	Premiers essais avec les Histogrammes de Gradients Orientés	13
	Résultats	14
5.3.2	Ajout d'une information contextuelle	14
	Protocole de test	14
6	Conclusion	16
7	Bibliographie	17
8	Annexes	18

1 Contexte

Le projet GéoPeuple est né d'un intérêt commun entre l'École des Hautes Études en Sciences Sociales (EHESS), le Laboratoire d'informatique de Paris 6 (LIP6) et l'Université Pierre et Marie Curie (UPMC). La Bibliothèque nationale dispose d'une multitude de données cartographiques sous forme de cartes géographiques datant des siècles passés. Ces cartes sont extrêmement riches en information de cette époque. Ainsi, elles montrent la toponymie des lieux de l'époque, la répartition et la manière dont le territoire était habité à cette période, le réseau orohydrographique et même la localisation de certains faits et sites historiques. Il s'avère donc intéressant d'enrichir et d'exploiter ces cartes notamment avec une géolocalisation des différents objets présents sur les cartes.

Le projet GéoPeuple vise à étudier les dynamiques des populations à travers l'observation du territoire dans le temps. L'équipe MaLire (Machine Learning and Information Retrieval) du LIP6 (Matthieu Cord, Nicolas Thome, Thierry Artières, Jonathan Guyomard) participe à ce projet avec pour objectif de fournir des algorithmes qui permettront l'automatisation de la vectorisation de cartes anciennes, ce qui facilitera la construction de bases de données historiques.

Les premières vectorisations seront effectuées sur des cartes de Cassini, tracées entre 1756 et 1815. Dans un premier temps, quatre zones ont été retenues pour être vectorisées manuellement afin d'établir et de tester les différents algorithmes : Saint Malo, Reims, Grenoble et Agen (Voir Fig 1)

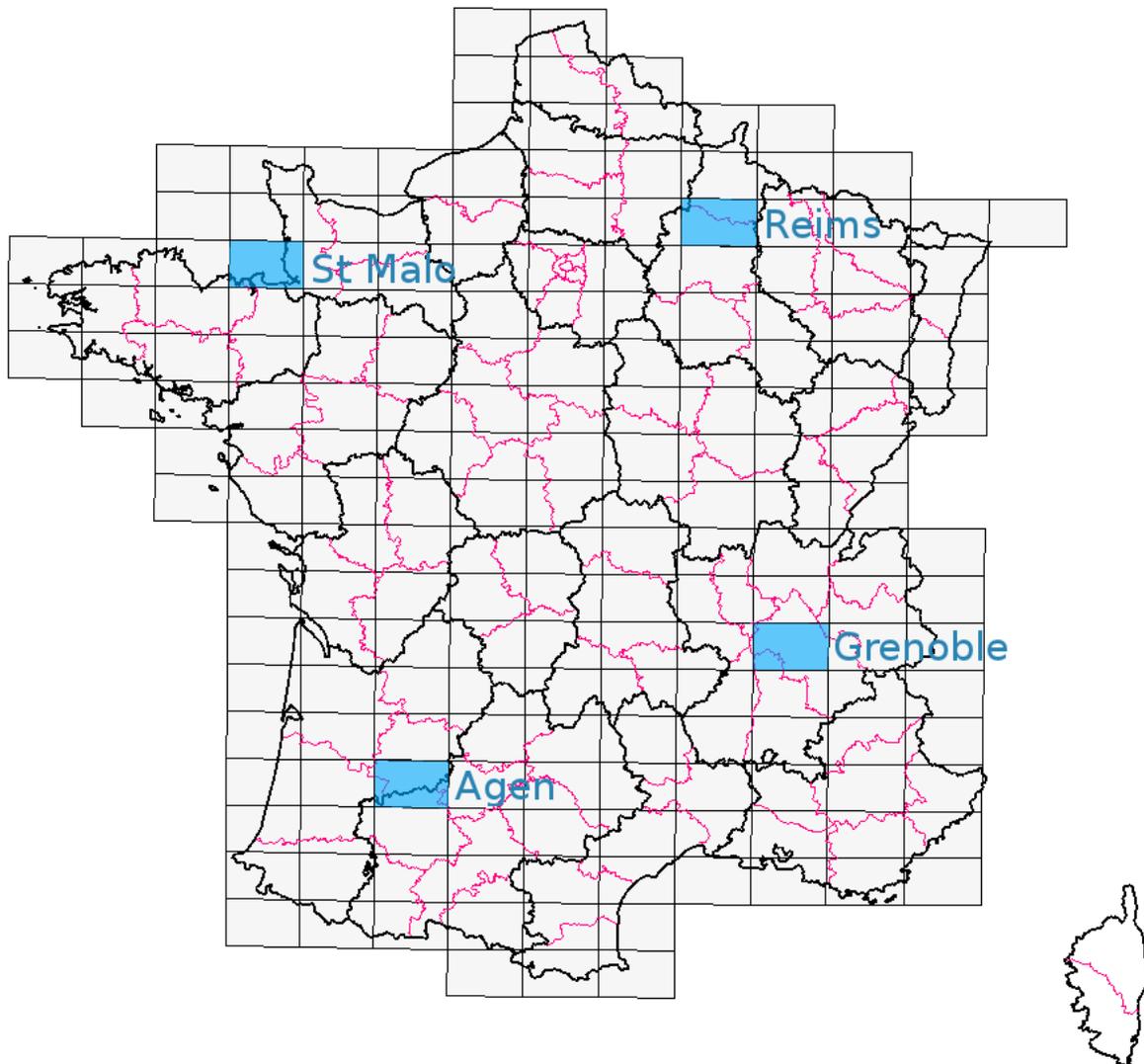


FIGURE 1 – Découpage de la France par les cartes de Cassini

2 Introduction

Ce rapport traite des travaux du LIP6 concernant la vectorisation automatique des cartes de Cassini. Il s'oriente autour de trois grandes parties, premièrement, une description générale de la méthode utilisée, contenant un état de l'art non exhaustif des descripteurs images utilisés ainsi que la présentation de l'algorithme d'apprentissage. Deuxièmement, une partie présentant les mesures utilisées pour évaluer nos algorithmes. Et enfin, le rapport se termine sur la présentation des premiers résultats de détection sur la carte de Reims.

3 Présentation générale de la méthodologie

En vision par ordinateur, un système de détection d'objets repose en général sur deux axes :

- l'utilisation de descripteurs images adaptés aux objets à détecter
- l'apprentissage supervisé par un outil d'apprentissage statistique

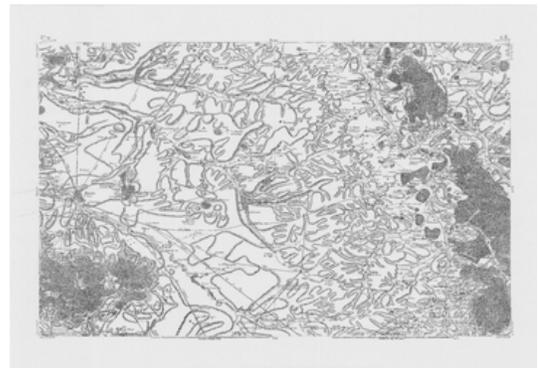
Dans cette partie, nous commençons par décrire la méthode classique utilisée pour la détection d'objets dans les images numériques, dites de fenêtre glissante. Ensuite, nous définissons le terme de descripteur d'images et présentons de manière non exhaustive ceux que nous avons testé ou utilisé. Et enfin, nous présentons l'algorithme utilisé pour effectuer l'apprentissage et la classification des données.

3.1 Spécifications des images

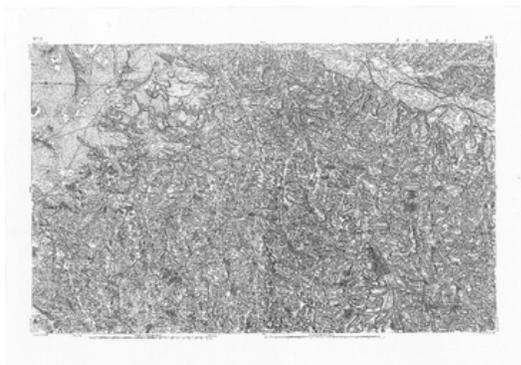
Les cartes de Cassini ont été numérisées et sont fournies par l'IGN. Il a été défini au préalable que l'annotation exhaustive des objets ponctuels se ferait sur quatre cartes (Saint Malo, Reims, Grenoble et Agen, voir Fig 2) afin de permettre l'extraction de base de données fiables servant à l'apprentissage. Chaque carte est fournie sous le format Tagged Image File Format (TIFF), avec une dimension d'environ 24000×17000 pixels. Les éléments ponctuels à détecter sont présentés Fig 26, et ont des dimensions allant d'une trentaine à une centaine de pixels en largeur et en hauteur.



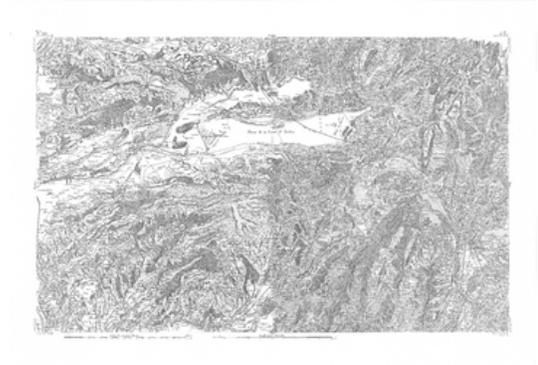
(a) Saint-Malo



(b) Reims



(c) Agen



(d) Grenoble

FIGURE 2 – Extrait de cartes de Cassini

3.2 Fenêtre glissante

La méthode de détection utilisée est une méthode dites de fenêtre glissante (sliding-window, Fig 3). La carte est parcourue avec un certain pas par une fenêtre de taille fixe. Pour chaque position de la fenêtre un descripteur (décrit partie 3.3) est calculé, et la fenêtre est, à l'aide d'un outil de classification (décrit partie 3.4), définie comme étant ou n'étant pas un objet recherché.

Afin d'avoir une détection exhaustive, le pas de balayage de la fenêtre doit être relativement faible, ce qui implique un très grand nombre de données à traiter et de calcul à effectuer. Pour une carte de 24000×17000 pixels, le nombre de fenêtres à traiter serait de :

- pas de 1 pixel : 408 millions de fenêtres
- pas de 2 pixels : 102 millions de fenêtres
- pas de 3 pixels : 45 millions de fenêtres

De plus, le nombre d'objets à détecter dans une carte étant plutôt faible, cela apportera lors de l'apprentissage un fort déséquilibre entre le nombre d'exemples positifs et le nombre d'exemples négatifs d'apprentissage, ce qui peut entraîner un apprentissage non optimal. Ce problème sera résolu avec l'utilisation, lors de l'apprentissage, d'algorithmes dit de hard-negatifs.

3.3 Descripteurs d'images

En vision par ordinateur et traitement d'images, le terme de descripteur (ou feature) est utilisé pour indiquer une information pertinente qui permettra de résoudre des calculs liés à certaines applications. L'extraction des descripteurs d'images consiste en des transformations mathématiques calculées sur les pixels d'une image numérique.

On distingue trois grandes familles de descripteurs :

- Descripteur de couleur
- Descripteur de texture
- Descripteur de forme

Étant donné nos besoins (détection d'objets ponctuels sur des cartes en noir et blanc), nous décrivons dans les chapitres qui suivent quelques uns des descripteurs de formes, basés sur l'extraction de l'information contenue dans les contours des symboles, que nous avons utilisé pour effectuer nos détections. Si on en venait à devoir détecter des régions (du type forêts, plaines, etc...), il serait intéressant d'effectuer une analyse plus poussée sur les descripteurs de texture, qui seraient alors plus informatifs pour ce type de tâche.

3.3.1 Distance de Chanfrein - Contours

La distance de Chanfrein [5] est utilisée dans le domaine de l'imagerie pour mesurer une distance (ou similarité) entre objets. La distance de Chanfrein calculée pour une image donne est représentée Fig 4.

On peut observer que les zones les plus sombres correspondent aux contours de l'image. Plus on s'éloigne du contour et plus la zone devient claire. Autrement dit, on a une valeur égale à zéro quand on est sur les contours, et plus on est loin du contours, plus la valeur augmente. Ainsi, on peut calculer la similarité entre deux images. Il suffit de calculer les contours sur la deuxième image et comparer chaque pixel du contours avec la zone correspondante sur la première.

Dans le cas idéal, c'est-à-dire quand on compare une image avec elle-même, la distance doit être égale à zéro.

Algorithme de Canny pour le calcul des contours



FIGURE 3 – Exemple d'une fenêtre glissante

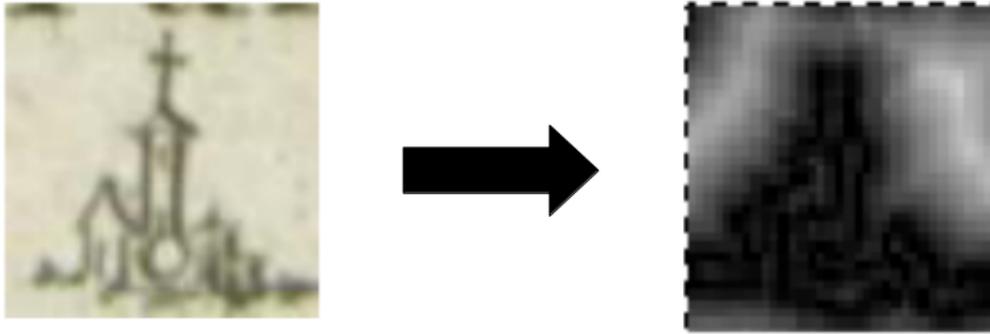


FIGURE 4 – Transformation d'une image avec la distance de chanfrein

Pour calculer les contours (voir Fig 5), on utilise l'algorithme de Canny [1]. On procède de la façon suivante :

1. Réduire le bruit de l'image (filtre Gaussien)
2. Calculer l'intensité des contours (calcul des gradients)
3. Calculer l'orientation des contours
4. Suppression des non-maxima
5. Seuillage des contours



FIGURE 5 – Exemples de calcul à l'aide de l'algorithme de Canny

3.3.2 Histogrammes de gradients orientés

Le but des Histogrammes de gradients orientés (HoG) [3] est de représenter l'apparence et la forme d'un objet dans une image grâce à la manière dont sont réparti l'intensité du gradient ou la direction des contours. Ceci est effectué en divisant l'image en cellules et en calculant pour chaque cellule un histogramme des directions du gradient pour les pixels appartenant à cette cellule. La concaténation de ces histogrammes forme le descripteur.

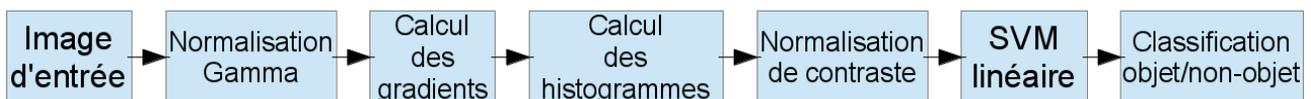


FIGURE 6 – Chaîne d'extraction des HoG

Normalisation Gamma/couleur

L'extraction du descripteur a été évalué avec plusieurs représentations couleur des pixels : niveau de gris, RGB et LAB, optionnellement avec une normalisation Gamma. Cette normalisation n'a qu'une faible incidence sur les performances et n'est donc pas obligatoire. Les espace de couleur RGB et LAB donnent des résultats comparables, alors que le niveau de gris réduit les performances. L'utilisation de l'information couleur est donc recommandée dans le cas où celle-ci est disponible.

Calcul des gradients

Les performances du détecteur sont sensibles à la manière dont ont été calculés les gradients, mais la technique la plus simple semble être la meilleure. Dalal et Triggs ont testé de pré-traiter l'image avec un lissage Gaussien, puis de calculer les gradients en appliquant des filtres dérivatifs différents (plusieurs σ ont été testés pour le lissage Gaussien) :

Filtre dérivatif 1-D non centré	$[-1 \ 1]$
Filtre dérivatif 1-D centré	$[-1 \ 0 \ 1]$
Filtre dérivatif 1-D ajusté cubiquement	$[1 \ -8 \ 0 \ 8 \ -1]$
Filtre de Sobel 3x3	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
Filtre diagonale 2x2	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$ et $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$

Les meilleurs résultats sont obtenus avec le filtre dérivatif centré (voir Fig 7), avec $\sigma = 0$ (aucun lissage). Pour les images couleurs, les gradients sont calculés séparément sur chaque canal couleur, le gradient ayant la norme la plus grande est gardé.

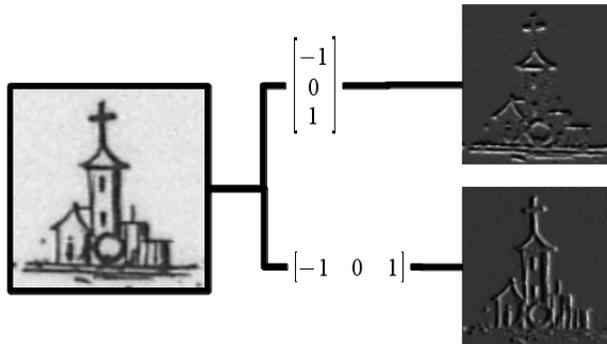


FIGURE 7 – Gradients calculés sur une vignette d'église avec des filtres dérivatifs

Calcul des histogrammes

L'image est découpée en plusieurs cellules de petite taille (Fig 8.a), et pour chaque cellule un histogramme est calculé. Chaque pixel d'une cellule vote pour une orientation entre 0 et 180 dans le cas non signé (Fig 8.b), ou entre 0 et 360 dans le cas signé.

L'étape suivante est la normalisation des descripteurs, afin d'éviter les disparités dues aux variations d'illumination, ainsi que l'introduction de redondance dans le descripteur. Pour cela, les cellules sont regroupées par bloc (concaténation des histogrammes des cellules d'un bloc), le vecteur de valeur du bloc est ensuite normalisé. Les blocs se recouvrent, donc une même cellule peut participer plusieurs fois au descripteur final.

Les normalisations possibles du descripteur final sont les suivantes (v représentant l'histogramme d'un bloc) :

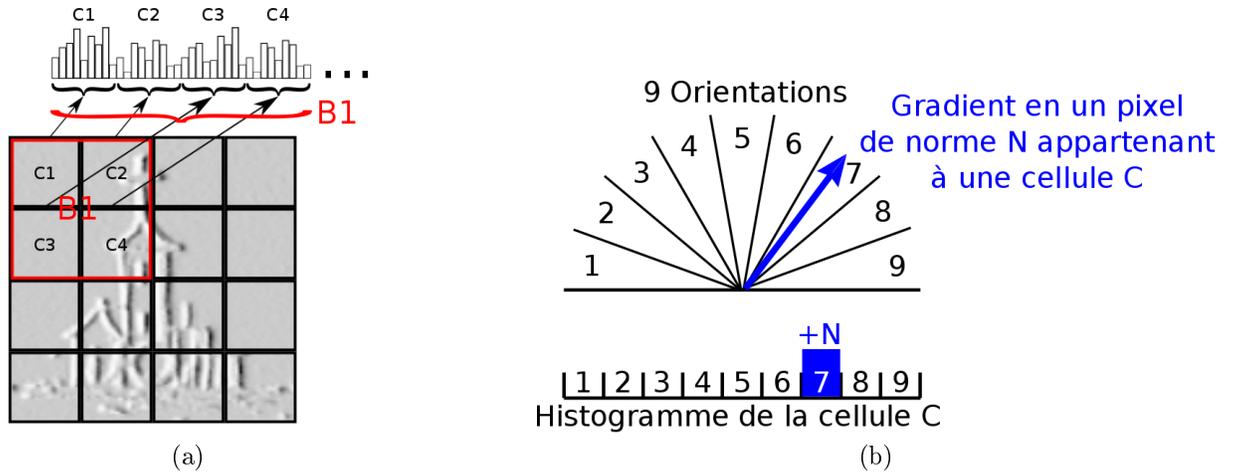


FIGURE 8 – Formation du HoG d'une image : (a) découpage en cellules et formation de l'historgramme par bloc, (b) vote d'un pixel selon l'orientation de ses gradients.

L2-Norme	$v \rightarrow \frac{v}{\sqrt{\ v\ _2^2 + \epsilon^2}}$
L1-Norme	$v \rightarrow \frac{v}{\ v\ _1 + \epsilon}$
L1-Racine	$v \rightarrow \sqrt{\frac{v}{\ v\ _1 + \epsilon}}$

Ainsi qu'une quatrième norme, L2-Hys, qui consiste à calculer v par la L2-norme, limiter les valeurs maximales de v à 0.2, puis renormaliser. Les normes L2-norme, L2-hys et L1-racine obtiennent des performances similaires, tandis que la L1-norme obtient de moins bons résultats, qui restent néanmoins meilleurs qu'une absence de normalisation.

3.3.3 Descripteur contextuel

Afin de résoudre certains problèmes liés à la première détection effectuée avec un descripteur d'images visuel traditionnel, nous avons décidé d'implémenter un vecteur de description contenant une information de contexte entre classe. Il arrive par exemple fréquemment qu'un calvaire soit détecté au sommet d'une église (voir Fig 9).

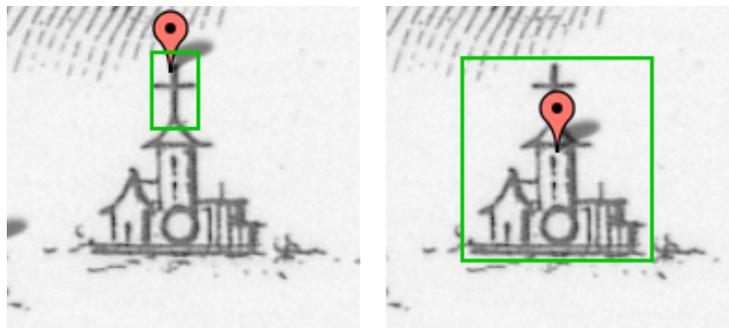


FIGURE 9 – Cas récurrent de mauvaise détection d'un calvaire (une simple croix) au sommet d'une église

Rajouter une information de contexte pourrait :

- Valider fortement la détection d'une église si un calvaire est détecté à son sommet
- Rejeter un calvaire détecté si une église est détectée en dessous de celui-ci

Ce cas étant le plus flagrant, il est probable que des informations de contexte lient d'autres classes entre elles.

Calcul du descripteur

Le vecteur de description d'une fenêtre pour la classe C_i est défini de cette manière :

$$[Score(C_i), \max(Score(C_1)), position(C_1), \dots, \max(Score(C_n)), position(C_n)]$$

Avec :

- La fonction $Score(C_i)$ étant la valeur de sortie du classifieur entraîné avec les HoG.
- La fonction $position(C_i)$ donnant la position (soit la distance, soit le vecteur position $[x, y]$) de la fenêtre ayant le score maximum pour la classe C_i dans le voisinage de la fenêtre en cours.

Afin d'évaluer les performances d'un descripteur de contexte de ce type, il est nécessaire d'avoir un protocole clair d'extraction d'une base de données représentant correctement chacune des classes.

Analyse qualitative/quantitative

Dans un premier temps, il est possible d'effectuer une analyse qualitative du contexte en se servant uniquement de la vérité terrain pour définir nos exemples positifs et remplir nos vecteurs de description. Le vecteur de description d'une fenêtre est défini de cette manière :

$$[Prsence(Score(C_1)), \min(C_1), \dots, Prsence(Score(C_n)), \min(C_n)]$$

Avec :

- La fonction $Prsence(C_i)$ indiquant si oui (valeur à 1) ou non (valeur à -1 ou 0) la classe C_i est présente dans le voisinage de la fenêtre en cours.
- La fonction $\min(position(C_i))$ indiquant, en cas de présence de la classe C_i dans le voisinage, la position la plus proche de celle-ci par rapport à la fenêtre en cours.

Cas concret

Dans un second temps, il faut réussir à construire les vecteurs de description à l'aide du détecteur de premier niveau appris sur les HoG. Dans ce cas, deux possibilités se présentent :

- N'utiliser que les boîtes de détection au premier niveau ayant un score supérieur à un seuil S pour former la base de données :
 - Positifs : l'ensemble de la vérité terrain.
 - Négatif : les fenêtres ayant un score supérieur à S et qui ne se superposent pas avec la vérité terrain.
- Utiliser l'ensemble des boîtes de détection au premier niveau et partir sur une stratégie hard-négatifs comme pour l'apprentissage du classifieur au premier niveau.

3.4 Apprentissage supervisé

Dans le domaine de l'intelligence artificielle il existe différents types d'apprentissage artificiel. L'apprentissage supervisé utilise une base d'apprentissage contenant des exemples annotés. A partir de cette base de départ et de ces annotations, le système est capable de classer un nouvel exemple qui n'appartient pas à la base. Dans notre cas, nous utilisons un classifieur qui prends des décisions binaires, c'est-à-dire "l'objet appartient à la classe ou l'objet n'appartient pas à la classe". Les bases de données seront donc constituées d'exemples positifs (par exemple "église"), et d'exemples négatifs (tout le reste : forêts, champs, montagnes, autres classes ponctuelles,...).

3.4.1 Outil d'apprentissage : Support Vector Machin (SVM)

Dans le cadre de l'apprentissage supervisé on a utilisé des machines à vecteurs de support (Support Vector Machine) [2]. Ce sont des classifieurs qui permettent de traiter des problèmes de discrimination de données. D'abord on dispose de deux ensembles, l'un avec les exemples positifs (motifs que l'on veut trouver) et l'autre avec les exemples négatifs (tout ce qui n'est pas le motif recherché). Le but est de trouver un hyperplan séparateur qui maximise la distance entre ces deux ensembles, Fig 10.

Une fois l'axe calculé, on souhaite classer un nouvel exemple n'appartenant pas à la base de départ. Pour ce faire, on projette cet exemple sur l'axe discriminant. Dans le cas où les deux ensembles ne sont pas linéairement séparables, il faut réaliser un changement de dimension, à l'aide d'une fonction non-linéaire φ . Après ce changement, on effectue une séparation linéaire.

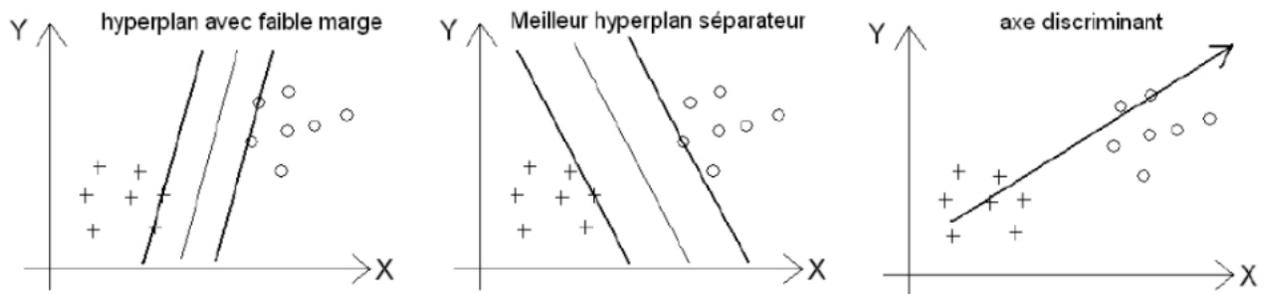


FIGURE 10 – Différents hyperplans permettant la séparation des deux groupes de données

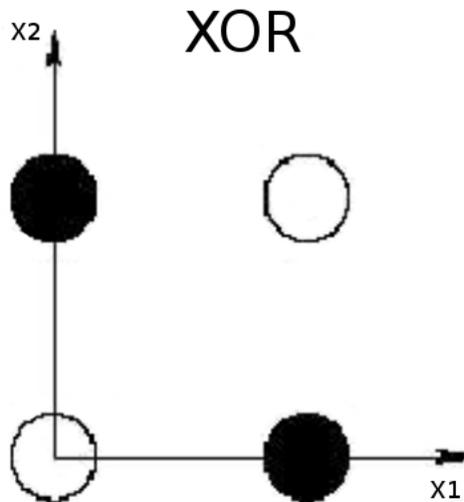


FIGURE 11 – Fonction XOR

Le cas de la Fig 11 n'est pas linéairement séparable en deux dimensions.

Mais si on utilise la fonction $\varphi(x, y) \rightarrow (x, y, x \times y)$ qui transforme l'espace en trois dimensions, alors la séparation est possible (Fig 12).

En pratique, au lieu de trouver une fonction $\varphi : X \rightarrow F$ qui transforme l'espace, on choisit une fonction $k : X \times X \rightarrow R$ appelée fonction noyau. Elle représente un produit scalaire dans l'espace de représentation intermédiaire. Elle traduit donc la répartition des exemples dans cet espace : $k(x, x') = \varphi(x) \cdot \varphi(x')$

Exemples de noyaux :

- Linéaire : $k(x, x') = xx'$
- Polynomial : $k(x, x') = \frac{(xx')^d - \|x - x'\|^2}{\sigma}$
- Gaussien : $k(x, x') = e^{-\frac{\|x - x'\|^2}{\sigma}}$

3.4.2 Gestion des hard-négatifs

La première étape afin d'effectuer notre apprentissage est d'extraire une base de données représentative de ce que l'on doit détecter. Dans notre cas, l'IGN nous a fourni un ensemble de fichiers contenant les coordonnées des boîtes englobantes de chaque objet de chaque classe dans les cartes. A partir de ces fichiers, l'extraction de la base de données s'effectue ainsi :

- Positifs extraits : Fenêtres annotées par l'IGN.
- Négatifs extraits : Sous-échantillonnage (scan avec un pas de 10–20 pixels au lieu de 1–2 pixels) d'un balayage complet de la carte sans fenêtre qui se superpose avec la vérité terrain.

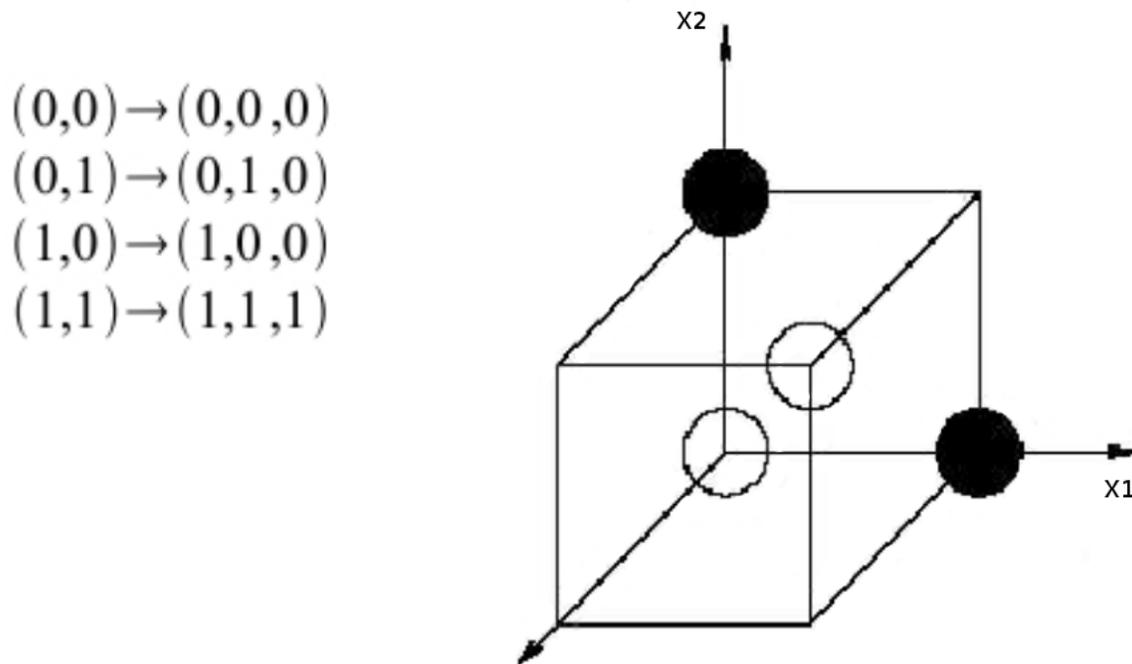


FIGURE 12 – Fonction XOR transformée

On obtient ainsi une base de données pour chaque classe à détecter (la fenêtre englobante moyenne de chaque classe étant plus ou moins grande), avec le problème cité précédemment d'un fort déséquilibre entre positifs et négatifs. Le ratio positifs/négatifs est en moyenne de 1/5000.

L'algorithme de hard-négatifs est une stratégie de sélection d'exemples d'apprentissage. En effet, certains exemples s'avèrent beaucoup plus difficile à classer que d'autres. Ces exemples, appelés hard-négatifs vont constituer une nouvelle base d'apprentissage.

On utilise une méthode de sélection décrite dans [4] :

```

Choisir it le nombre d'itération maximum, r le ratio négatifs/positifs
P ensemble des positifs de taille p, N ensembles de négatifs de taille n
S0 = sous-ensemble aléatoire de N de taille p*r
C classifieur à entraîner
i = 0
Tant que i < it ou Si ≠ Si-1
  - entraîner C avec P et Si
  - calculer les scores des exemples de N avec C
  - Si+1 = p*r exemples les moins bien classés de N
  - i = i + 1

```

4 Évaluation des performances

Pour comparer les différents algorithmes de classification, il est impératif de définir une mesure pour évaluer leurs performances. Nous présentons dans cette partie une mesure des performances fréquemment utilisée par la communauté de vision par ordinateur, qui a l'avantage de ne pas être influé par le déséquilibre entre exemples positifs et exemples négatifs que l'on rencontre dans notre cas.

4.1 Mesures de la précision et du rappel

On mesure l'efficacité d'une technique de recherche d'informations en utilisant deux mesures distinctes, la précision et le rappel. Ces deux quantités sont définies lorsque l'algorithme de décisions retourne une valeur binaire : soit un document appartient à la classe que l'on recherche, soit il n'y appartient pas. Dans un tel cas, on peut évaluer les quantités suivantes décrites dans le tableau 4.1.

	Positifs vérité terrain	Négatifs vérité terrain
Classés positifs	Vrais positifs	Faux positifs
Classés négatifs	Faux négatifs	Vrais négatifs

Le rappel et la précision sont alors respectivement définis par les formules (1) et (2).

$$R = \frac{\text{Nombre de vrais positifs}}{\text{Nombre de vrais positifs} + \text{Nombre de faux négatifs}} \quad (1)$$

$$P = \frac{\text{Nombre de vrais positifs}}{\text{Nombre de vrais positifs} + \text{Nombre de faux positifs}} \quad (2)$$

Le rappel est le rapport du nombre de vrais positifs correctement classés sur le nombre total de positifs. Cela correspond au pourcentage de réponses correctes qui sont données. La précision est la proportion de vrais positifs parmi l'ensemble des positifs retournés par le classifieur. Cela correspond au pourcentage de réponses correctes.

En pratique, il est facile de fournir un système avec un rappel de 100%, il suffit que le classifieur ne retourne que des positifs. C'est également le cas pour obtenir une précision de 100%, il suffit que le classifieur quasiment aucun positifs, à part quelques uns dont la confiance est assurée. En réalité, le but est de trouver le meilleur compromis entre le rappel et la précision. Un classifieur parfait doit avoir un rappel et une précision de 100%.

4.2 Précision moyenne (Average precision)

Comme expliqué précédemment, la précision et le rappel doivent être considérés simultanément pour évaluer et comparer différents algorithmes de détection. Une mesure répandue qui prends en compte à la fois le rappel et la précision est la précision moyenne (average precision). On exprime la précision comme une fonction du rappel : $p(r)$. La valeur moyenne de $p(r)$ sur l'ensemble de l'intervalle $r = 0$ à $r = 1$ est appelé précision moyenne :

$$AP = \int_0^1 p(r) dr$$

En pratique, l'intégrale est remplacée par une somme finie :

$$AP = \int_0^1 p(r) dr = \sum_{i=1}^n p(i) \Delta r(i)$$

où $\Delta r(i)$ est le changement de rappel entre $i - 1$ et i .

5 Expérimentations et résultats

5.1 Création d'une interface graphique

Il est évident qu'une application qui traite avec des bases d'images a besoin d'une interface graphique qui permette de visualiser les résultats d'une détection afin d'effectuer une appréciation visuelle qualitative rapide. Cette interface a été développée à l'aide de l'API GoogleMaps, et permet de visualiser rapidement dans un navigateur web les résultats d'une détection (Fig 24 et Fig 25).

5.2 Intérêt des hard-négatifs

Afin de déterminer l'intérêt d'effectuer l'apprentissage de nos classifieurs avec une stratégie de hard-négatifs, plusieurs tests ont été effectués en faisant varier la taille du sous-ensemble de négatifs (tableau 14) d'une base de données (tableau 13). Les tests sont effectués avec un classifieur SVM pour 5 itérations de l'algorithme.

Catégorie : Moulin à eau	
Base de TRAIN	Base de TEST
Positifs : 80	Positifs : 102
Négatifs : 488756	Négatifs : 488368

FIGURE 13 –

Pourcentage de la base de TRAIN utilisé	Nombre d'exemples négatifs	MAP TRAIN	MAP TEST
100%	488756	10^{-4}	10^{-4}
50%	244378	10^{-3}	10^{-3}
25%	122189	0.04	0.04
10%	48875	0.839	0.724
5%	24437	0.918	0.858
2%	9775	0.952	0.902
1%	4887	0.975	0.912
0.17%	830	0.990	0.916
0.085%	415	0.06	0.04
0.034%	166	0.02	0.01
0.017%	83	0.04	0.04

FIGURE 14 –

Comme on peut le voir avec ces résultats, lorsque l'on utilise la base complète pour entraîner notre classifieur, celui-ci n'arrive pas à converger vers une solution qui sépare correctement les données. De même lorsque le pourcentage de négatifs est trop faible, les exemples négatifs ne sont alors plus assez représentatifs par rapport à la quantité totale des négatifs de la base de données.

Dans notre cas, et si l'on se fie au tableau ci-dessus, il suffit d'environ un millier d'exemples négatifs pour que l'apprentissage du classifieur se déroule de manière quasiment optimale. Pour les prochains tests, nous partiront sur cette base et utiliseront un sous-ensemble d'environ mille exemples négatifs pour entraîner nos classifieurs.

5.3 Détection des objets ponctuels

5.3.1 Premiers essais avec les Histogrammes de Gradients Orientés

Les premières expérimentations ont été effectuées sur la carte de Reims. Le but était de détecter seize classes différentes d'objets ponctuels, mais le peu d'exemples positifs disponibles pour certaines d'entre elles ne permet d'obtenir des résultats que sur neuf de ces catégories.

Les premiers tests sont effectués sur la carte de Reims, celle-ci est découpée en deux parties afin d'extraire une base d'apprentissage et une base de test (Fig 15). L'extraction des négatifs se fait avec un pas de 20 pixels, ce qui donne des bases de données contenant pour chaque classe environ 500 000 exemples négatifs. Les tests ont été effectués avec deux implémentations différentes du SVM (Tableaux 16 et 17).

Le nombre de négatifs utilisés varie en fonction des catégories, la convergence n'est pas assurée dans tout les cas pour environ 1000 exemples.

L'implémentation DoubleSGD sera dorénavant utilisée, les temps de calcul étant plus court pour des résultats relativement similaires.

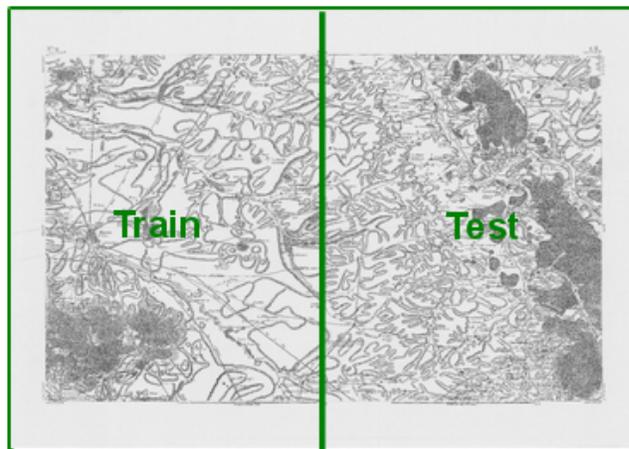


FIGURE 15 – Découpage de la carte de Reims en deux sous-parties

Type	MAP Train	MAP Test	Positifs en train	Positifs en test
Source	0.92	0.39	23	71
Église	0.88	0.66	167	118
Chapelle	0.99	0.05	15	8
Calvaire	0.87	0.39	25	30
Hameau	0.85	0.46	43	75
Château	0.86	0.26	37	63
Écart	0.94	0.74	31	96
Moulin à eau	0.99	0.91	80	102
Moulin à vent	0.93	0.77	28	19

FIGURE 16 – Classifieur : DoublePegasosSVM, Descripteur : HoG, Hard negatives : 1000

Type	MAP Train	MAP Test	Positifs en train	Positifs en test
Source	1.0	0.39	23	71
Église	0.84	0.57	167	118
Chapelle	1.0	0.05	15	8
Calvaire	0.99	0.24	25	30
Hameau	0.84	0.27	43	75
Château	0.75	0.12	37	63
Écart	1.0	0.67	31	96
Moulin à eau	1.0	0.85	80	102
Moulin à vent	1.0	0.64	28	19

FIGURE 17 – Classifieur : DoubleSGD, Descripteur : HoG, Hard negatives : 1000

Résultats

Les figures 27, 28, 29 sont des résultats visuels issue d'une détection effectuée avec les HoG.

5.3.2 Ajout d'une information contextuelle



FIGURE 18 – Le rectangle vert correspond à une détection visuelle d'un calvaire, qui est en réalité le sommet d'une église, donc un faux positif. Le rectangle rouge correspond à une détection visuelle d'une église, qui elle, est correcte. Le cercle bleu correspond à la zone où l'on va observer le contexte.

Dans certains cas, l'information visuelle ne sera pas suffisante pour permettre de classifier correctement certains objets (Fig 19). Afin de corriger ce problème, on ajoute une information contextuelle à notre descripteur, comme illustré Fig 18. Dans le cas présent, le descripteur sera représenté par le score de la détection visuelle du calvaire, ainsi que le score de la détection visuelle de l'église et sa position relative. Cet exemple sera annoté comme étant un négatif, lors d'un apprentissage sur ces nouveaux descripteurs, les détections de calvaires au sommet des églises devraient alors être filtrées.

Protocole de test

Afin de tester la validité du descripteur de contexte, les premiers tests seront effectués en utilisant seulement deux classes dont on est sûr qu'il y ait un information contextuelle entre elle, c'est à dire les calvaires et les églises.

- Pour ces tests, la carte de Reims est découpée en trois parties afin d'obtenir une base d'apprentissage, une base de test et une base de validation.
- On entraîne un classifieur de premier niveau, pour chaque classe, en utilisant les histogrammes de

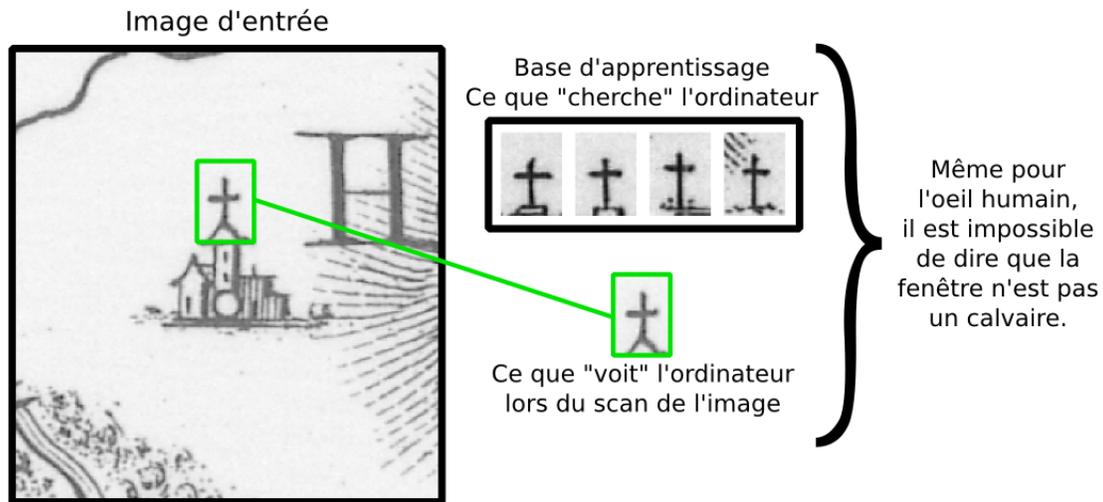


FIGURE 19 – Limitations d'une détection purement visuelle

gradients orientés (Classifieur : SGD, avec hard négatifs, on obtient une convergence de l'algorithme pour 6000 exemples négatifs, soit 2% de la base de négatifs complète).

- Les seuils de détection sont légèrement modifiés afin d'autoriser la détection des fenêtres les plus ambiguës (qui devraient être les croix au sommet des églises dans le cas de la classe calvaire notamment).
- Les descripteurs de contexte sont extraits et un nouveau classifieur est entraîné (Classifieur : SGD)

On peut observer que ces cas apparaissent dans la base de données directement de manière visuelle (Fig 20) ou lors de l'extraction des descripteurs de contexte des églises (Fig 21). En cas de non détection d'une classe dans le voisinage proche (pour le moment fixé à un rayon de 70 pixels), les valeurs du score de classification, de dx et de dy sont mises à -10 . On observe alors un gain d'environ 10% sur le MAP de la base de test, et d'environ 13% sur le MAP de la base de validation pour les calvaires. Pour les églises, le MAP diminue de quelques pourcent. De manière qualitative, on peut voir Fig 23 que les calvaires détectés à tort au sommet des églises sont correctement filtrés.

	Détections église	Détections calvaire
Exemple de la base d'apprentissage		
Exemple de la base de test		

FIGURE 20 – Exemples de détections qui confirment la présence d'exemples informatifs dans la base de train et de test



FIGURE 21 – Descripteurs contenus dans la base de test

MAP	HoG		Contexte	
	Église	Calvaire	Église	Calvaire
Base de train	0.875	1.0	0.861	1.0
Base de test	0.617	0.282	0.596	0.373
Base de validation	0.603	0.125	0.568	0.253

FIGURE 22 – Score de classification des églises et calvaires avec le HoG et avec le contexte

6 Conclusion

Le descripteur visuel utilisé donne déjà de très bon résultats étant donné la variabilité et le peu d'exemples positifs pour chaque symbole, et l'ajout d'une information contextuelle donne des résultats encourageants. Les perspectives pour la vectorisation automatique des cartes de Cassini sont, premièrement, d'utiliser un protocole de test plus robuste (nombre d'exemples dans chaque base, cross-validation), et deuxièmement, de rechercher dans la littérature des méthodes de description contextuelle totalement dédiées au contexte entre objets ou au contexte cartographique.

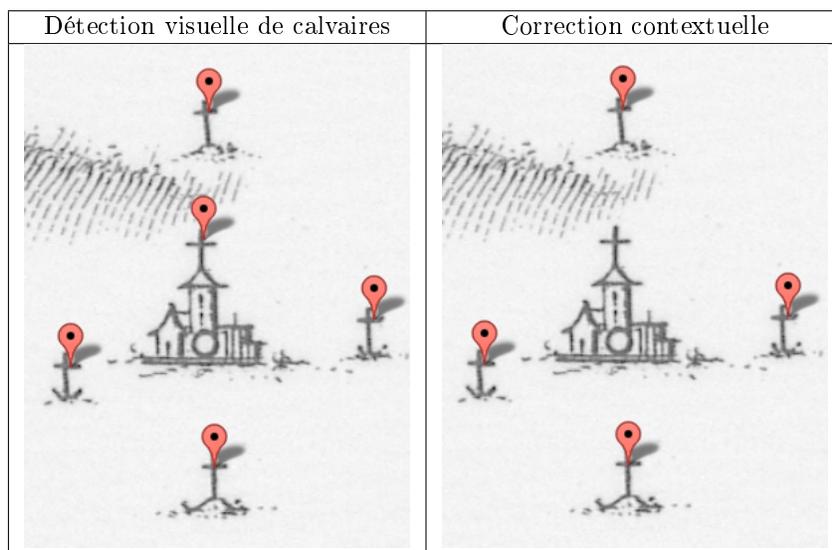


FIGURE 23 – Score de classification des églises et calvaires avec le HoG et avec le contexte

7 Bibliographie

Références

- [1] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6) :679–698, nov. 1986.
- [2] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20 :273–297, 1995. 10.1007/BF00994018.
- [3] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893, 2005.
- [4] Pedro Felzenszwalb, Ross Girshick, David Mcallester, and Deva Ramanan. Object detection with discriminatively trained part based models. Technical report, 2009.
- [5] E. Thiel and A. Montanvert. Les distances du chanfrein en géométrie discrète. In *Distancia'92, conf. int. sur l'analyse en distance*, pages 295–298, Rennes, Juin 1992.

8 Annexes

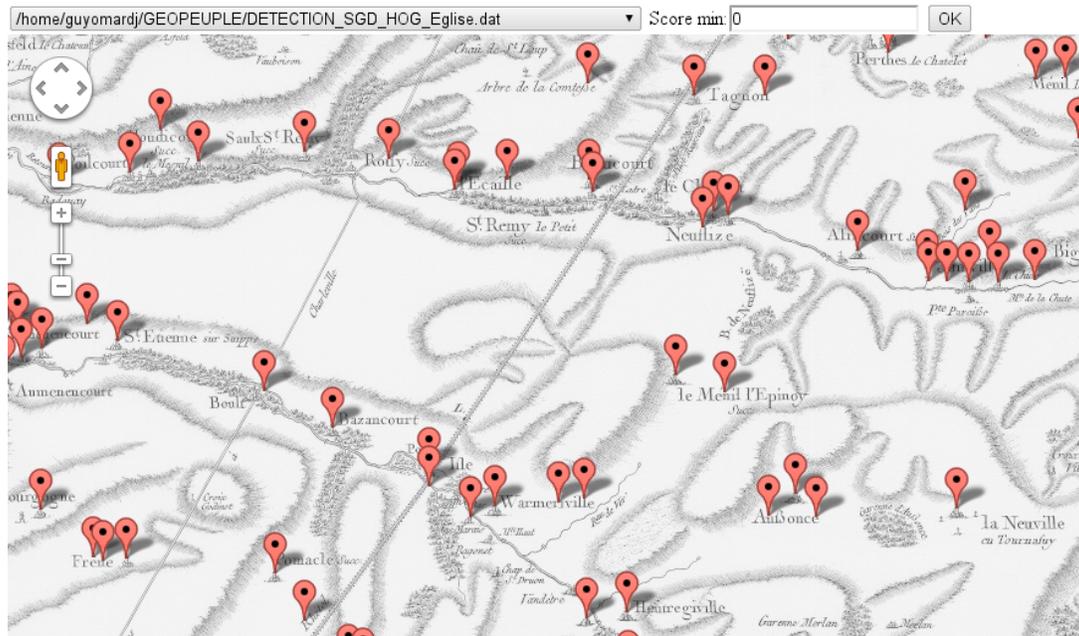


FIGURE 24 – Exemple du résultat d'une détection d'églises

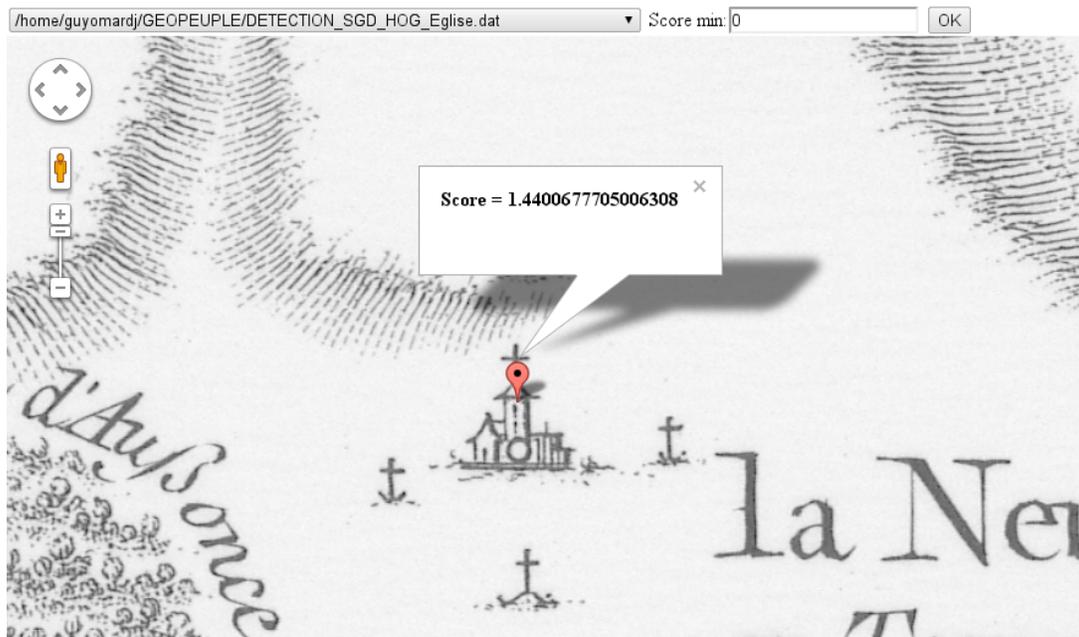


FIGURE 25 – Affichage du score de confiance d'une détection

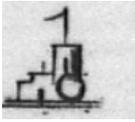
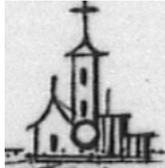
			
Source 94 exemples	Moulin à eau 182 exemples	Forge 3 exemples	Mine 6 exemples
			
Moulin à vent 47 exemples	Château 100 exemples	Corps de garde 1 exemples	Écart 127 exemples
			
Gentilhommière 16 exemples	Hameau 118 exemples	Abbaye 8 exemples	Calvaire 55 exemples
			
Chapelle 23 exemples	Commanderie 3 exemples	Église 285 exemples	Prieuré 5 exemples

FIGURE 26 – Les catégories soulignées sont celles ne donnant pour le moment aucun résultat correct de part le manque d'exemples



FIGURE 27 – Résultats de la détection de calvaires sur la carte de Reims



FIGURE 28 – Résultats de la détection d'églises sur la carte de Reims

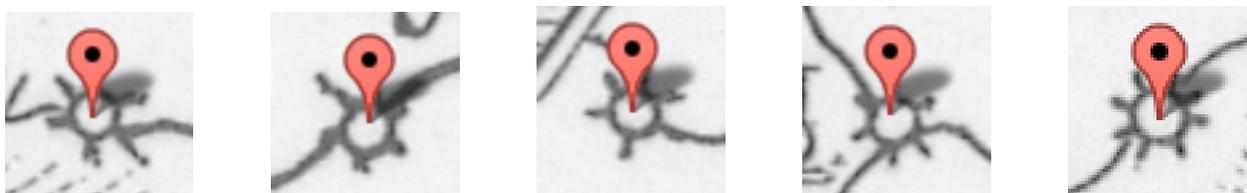


FIGURE 29 – Résultats de la détection de moulins à eau sur la carte de Reims



FIGURE 30 – Résultats de la détection d'écarts sur la carte de Reims



FIGURE 31 – Résultats de la détection de moulins à vent sur la carte de Reims