# Projet de Nuage de Points et Modélisation 3D: Fast Semantic Segmentation of 3D Point Clouds with Strongly Varying Density

Yannick Terme
yannickterme@gmail.com

20 Mars 2018

## 1 Introduction

La tâche de classification de points sur un nuage de points 3D brut peut être utilisée pour d'autres tâches comme l'extraction de surfaces ou d'objets. L'article Fast Semantic Segmentation of 3D Point Clouds with Strongly Varying Density, Hackel et Al [2] propose une méthode de classification point par point. Pour ce faire, on extrait dans un premier temps un ensemble de descripteurs des propriétés géométriques de chaque point puis un classifieur classique pour en déduire les probabilités d'appartenance à chaque classe. L'article se targue de 2 apports principaux: premièrement, la méthode est applicable à tous types de données, en particulier elle est robuste aux changements de densité; deuxièmement, la méthode est efficace d'un point de vue computationnel comparée aux autres méthodes. Pour les bases de données *Paris-Rue-Madame* et *Paris-Rue-Cassette* atteignent des précisions de 95-98% et des rappels de 93-99%.

Pour l'étude de cet article, dans un premier temps, j'ai implémenté les méthodes proposées par les auteurs et ai observé le score réalisé sur la base de données des rues de Lille [1]. Puis j'ai implémenté des modules qui améliorent les résultats.

## 2 Méthodes

Pour construire des descripteurs dont les capacités de classification sont robustes aux changement d'échelles, une des méthodes prépondérantes jusqu'alors était de calculer des descripteurs locaux sur une échelle adaptative [3], c'est-à-dire sélectionner le voisinage k-NN ou $\epsilon$-NN puis extraire les descripteurs associés. Dans ce cas, $k$ et $\epsilon$ s'adaptent à la densité de points. Le problème de cette

[1] cf. le dossier data. C'est aussi la base de données utilisée pour le TP4.

méthode est que dès que $k$ ou $\epsilon$ deviennent grands, le temps de calcul explose. Ainsi, les auteurs de [2] proposent une méthode de sous-échantillonnage du nuage de points qui permet de réduire énormément le temps de calcul. Grâce à ce gain, on peut calculer les descripteurs pour des échelles diverses tout en maintenant largement l'avantage compétitif en terme de calcul. Cette méthode montre aussi des performances de classification meilleures que les meilleures méthodes adaptatives telles que [3].

## 2.1 Approximation du voisinage

Les auteurs proposent d'utiliser un voisinage kNN car il s'adapte naturellement à des densités variables. A l'opposé, le voisinage $\epsilon$ ne s'adapte pas. La méthode de recherche des plus proches voisins est kDTree qui permet une recherche en $\mathcal{O}(log(n))$ plutôt qu'en $\mathcal{O}(n)$. Notons qu'il existe des méthodes kDTree approximatives qui réduisent le temps de calcul mais que nous n'utiliserons pas contrairement à l'article ce qui pourra en partie expliquer des différences de temps de calcul.

La méthode de calcul des descripteurs multi-échelles est la suivante:

1. sous-échantillonner le jeu de données sur plusieurs échelles d'une manière pyramidale. Une échelle est caractérisée par une taille de voxel. Sous-échantillonner avec une taille de voxel $v$ signifie quadriller l'espace en voxels de tailles $v$ et les points générés sont les moyennes des points de chaque voxel (en fait ce sont les médoïdes c'est-à-dire le point observé le plus proche de la moyenne)

2. à chaque échelle, extraire le voisinage: les $k$ plus proches voisins, où $k$ est faible et fixé. Puis calculer les descripteurs à chaque échelle.

**Choix des échelles** On s'intéresse en général à des objets qu'on peut trouver dans une rue tels que le sol, les bâtiments, les voitures, les panneaux, la végétation etc. dont les échelles de variation varient du centimètre au mètre. Les auteurs choisissent donc 9 échelles croissantes en partant de voxels de côtés 2.5 cm, et en sous échantillonnant d'un facteur 2 à chaque fois. Les tailles sont alors 2.5 cm, 5cm, 10 cm ... 6.4m.

## 2.2 Extraction des descripteurs

Une fois un voisinage déterminé, on calcule les valeurs et vecteurs propres du voisinage centré en 0. Les valeurs propres sont indicées par ordre décroissants $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$. Les valeurs propres sont normalisée à 1, de sorte à rendre les descripteurs robustes aux changements de densité. Les descripteurs sont classés en 3 groupes: covariance, moments, et hauteurs. Le 3ème groupe est particulièrement utile pour la description d'objets qui s'étendent sur la hauteur comme les arbres ou les lampadaires.

| | | |
|---|---|---|
| covariance | Sum | $\lambda_1 + \lambda_2 + \lambda_3$ |
| | Omnivariance | $(\lambda_1 \cdot \lambda_2 \cdot \lambda_3)^{\frac{1}{3}}$ |
| | Eigenentropy | $-\sum_{i=1}^{3} \lambda_i \cdot \ln(\lambda_i)$ |
| | Anisotropy | $(\lambda_1 - \lambda_3)/\lambda_1$ |
| | Planarity | $(\lambda_2 - \lambda_3)/\lambda_1$ |
| | Linearity | $(\lambda_1 - \lambda_2)/\lambda_1$ |
| | Surface Variation | $\lambda_3/(\lambda_1 + \lambda_2 + \lambda_3)$ |
| | Sphericity | $\lambda_3/\lambda_1$ |
| | Verticality | $1 - \lvert\langle[0\,0\,1], \mathbf{e}_3\rangle\rvert$ |
| moment | $1^{\text{st}}$ order, $1^{\text{st}}$ axis | $\sum_{i\in\mathcal{P}} \langle \mathbf{p}_i - \mathbf{p}, \mathbf{e}_1 \rangle$ |
| | $1^{\text{st}}$ order, $2^{\text{nd}}$ axis | $\sum_{i\in\mathcal{P}} \langle \mathbf{p}_i - \mathbf{p}, \mathbf{e}_2 \rangle$ |
| | $2^{\text{nd}}$ order, $1^{\text{st}}$ axis | $\sum_{i\in\mathcal{P}} \langle \mathbf{p}_i - \mathbf{p}, \mathbf{e}_1 \rangle^2$ |
| | $2^{\text{nd}}$ order, $2^{\text{nd}}$ axis | $\sum_{i\in\mathcal{P}} \langle \mathbf{p}_i - \mathbf{p}, \mathbf{e}_2 \rangle^2$ |
| height | Vertical range | $z_{\text{max}} - z_{\text{min}}$ |
| | Height below | $z - z_{\text{min}}$ |
| | Height above | $z_{\text{max}} - z$ |

Figure 1: Les descripteurs utilisés. $\mathcal{P}$ est l'ensemble des voisins. $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ sont les vecteurs propres associés respectivement à $\lambda_1, \lambda_2, \lambda_3$ (capture d'écran de l'article).

On extrait 16 descripteurs par échelle et par point. En comptabilisant les 9 échelles, chaque point a 144 variables.

### 2.3 3D shape context

Les descripteurs utilisées ci-dessus sont des descripteurs de covariance mais ne permettent pas de caractériser la forme de la surface environnante pour des objets complexes, en particulier les bords. Les auteurs utilisent alors 2 types de descripteurs, appelés "shape context descriptors": *Shape Context 3D (SC3D)* [1] et *Signature of Histogram of Orientations (SHOT)*. J'ai implémenté le premier, SC3D, et je vais donc me concentrer sur celui-ci. Afin d'extraire les descripteurs SC3D, on divise le voisinage du point en fonction de la distance et des angles des voisins. Plus précisément, on définit $J+1$ divisions radiales $\{R_0, ..., R_J\}$, $K+1$ divisions angulaires d'élévation$\{\theta_0, ..., \theta_K\}$, $L+1$ divisions angulaires azimutales $\{\Phi_0, ..., \Phi_L\}$. On divise alors le voisinage du point en $J \times K \times L$ compartiments. Puis on compte le nombre de points $p_i$ qui tombent dans chaque compartiment de l'histogramme auxquels on assigne des poids $w(p_i)$ définis ainsi:

$$w(p_i) = \frac{1}{\rho_i \sqrt[3]{V(j,k,l)}} \tag{1}$$

où $\rho_i$ est la densité autour du point $p_i$, ie le nombre de points à une distance inférieure à $\delta$, avec $\delta$ fixé; et $V(j,k,l)$ le volume du compartiment $(j,k,l)$[2].

L'histogramme de sortie est pondéré et c'est cet histogramme qui est le vecteur des descripteurs SC3D. La figure 2 illustre le découpage qui permet d'arriver aux histogrammes.

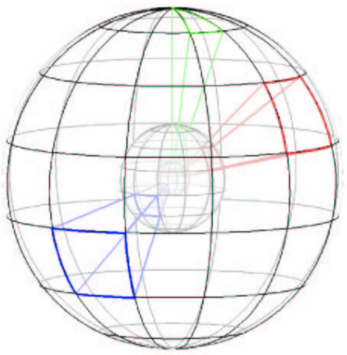

Figure 2: Visualisation des descripteurs SC3D issue de [1]

Les auteurs ont trouvé que ces descripteurs n'ajoutaient de l'information que pour les points qui se situaient à un bord. De plus ils l'ont rendu efficace en mêlant la structure pyramidale des niveaux d'échelle afin de calculer des approximations des descripteurs SC3D. Ils ont également créé un classifieur qui déterminait s'il y avait besoin de calculer les descripteurs SC3D. Pour ma part, je me suis contenté de calculer les descripteurs sans approximation et donc avec un coût de calcul beaucoup plus élevé.

En terme de résultat, comme les auteurs, je trouve une amélioration plutôt faible du score F1 moyen, dans mon cas, d'environ 1%. Mais je préfère utiliser les descripteurs des voisins significatifs, définis plus bas, car ils permettent également de caractériser le voisinage pour un prix moindre. Cependant, je n'ai pas réussi à faire usage des angles azimutaux. En effet, l'orientation de la rue est variable donc il faut orienter la sphère dans de manière intelligente, ce que je n'ai pas eu le temps de faire.

## 2.4 Classification

Le classifieur utilisé par l'article est un Random Forest. Un des avantages de cet algorithme est qu'il peut gérer un nombre important de variables, ce qui est le cas ici. Plutôt que de calculer les descripteurs sur la base de données entière, on échantillonne 1000 données de chaque classe. En plus de la réduction du temps de calcul, cela a l'avantage de ne pas discriminer les classes sous-représentées.

[2]https://en.wikipedia.org/wiki/Spherical_sector

## 2.5 Quelques ajouts

J'ai exploré des approches qui n'étaient pas abordées dans l'article. Parmi ces approches, je me suis permis de reprendre certaines méthodes qui avaient été présentées le jour de la présentation du mini-challenge et que j'avais trouvées judicieuses.

**Voisins significatifs** La première méthode consiste à calculer pour chaque point les descripteurs de voisins significatifs. Si l'on appelle $p$ le point considéré, ses voisins significatifs sont les points $p \pm \lambda_i \mathbf{e}_i$ avec $i \in \{1, 2, 3\}$. Cela porte le nombre de descripteurs à $144 + 144 \times 3 \times 2 = 1008$. Lors du calcul $p \pm \lambda_i \mathbf{e}_i$, on utilise les valeurs propres pas encore normalisées afin de prendre en compte la densité locale. Cependant, on utilise les valeurs propres normalisées lors du calcul des descripteurs. Ces descripteurs permettent d'obtenir un score F1 moyen légèrement plus élevé qu'avec les descripteurs simples.

**Lissage des prédictions par kNN** La méthode de lissage des prédictions. Un point qui a été classifié comme appartenant au sol alors que les points aux alentours ont été classifié comme appartenant à un bâtiment devrait être classifié comme bâtiment. Pour cela on utilise un kNN où les labels d'apprentissage sont les prédictions. Le désavantage de cette méthode est que des points à la frontière entre 2 entités seront très souvent corrigés, ce qui n'est pas souhaitable. Lors du calcul $p \pm \lambda_i \mathbf{e}_i$, on utilise les valeurs propres pas encore normalisées afin de prendre en compte la densité locale. Cependant, on utilise les valeurs propres normalisées lors du calcul des descripteurs. Ces descripteurs permettent d'obtenir un score F1 moyen légèrement plus élevé qu'avec les descripteurs simples.

**2ème méthode de lissage des prédictions** J'ai essayé de pousser le concept de lissage un peu plus loin après avoir remarqué que certains points avaient des voisinages absurdes et aussi afin de remédier au problème des points situés à la frontière entre 2 objets. L'idée est de prédire la classe d'un point à partir de son voisinage. Pour chaque point d'entraînement, on collecte les histogrammes des classes des points voisins et on prédit la classe du point en question à l'aide d'un Random Forest. Ainsi, si un lampadaire est prédit alors que son voisinage entier est un bâtiment et que le Random Forest attribue à cet évènement une probabilité en dessous de $t$, (où $t$ est un seuil défini à l'avance), alors le point est classifié d'après la prédiction du Random Forest. En fait, ce 2ème lissage supposé plus intelligent donne de mauvais résultats. On lui préfère le lissage simple par kNN. Les performances de classification peuvent alors être améliorées de 1% et elles sont optimisées avec un paramètre $k_{lissage} = 4$.

**Valeurs propres** Les valeurs propres étant normalisées pour sommer à 1, elles apportent donc une information qui ne dépend pas de la densité. On peut donc les ajouter comme variable.

**Régularisation par extraction de plans via RANSAC** Dans une rue d'une ville telle que celle que l'on étudie, on sait qu'il existe au moins 3 plans contenant uniquement des points appartenant à la même classe, à savoir la chaussée et les 2 façades des immeubles. Or j'ai remarqué via CloudCompare que des portions non négligeables des façades étaient parfois mal-classifiés alors qu'elles appartenaient au même plan que le reste de la façade, correctement classifiée. On peut alors extraire ces 3 plans avec l'algorithme RANSAC et assigner tous les points à la classe majoritaire de chaque plan. En pratique, on confirme que le premier plan extrait est le sol et les 2 autres sont 2 façades. Pour l'implémenter, j'ai utilisé l'algorithme `recursive_RANSAC` implémenté dans le TP6. J'ai choisi une largeur de plan de 10 cm, et le score F1 moyen est amélioré de 1 à 2%.

# 3 Résultats

**Données** Le classifieur est entraîné sur le jeu de données des 3 rues de Lille. La raison pour laquelle je ne suis pas allé jusqu'à optimiser de classifieur sur les jeux de données *Paris-Rue-Madame* et *Paris-Rue-Cassette* est computationnelle et je ne peux donc malheureusement pas comparer mes résultats à ceux obtenus dans l'article.

**Classifieur** Afin d'être sûr de ne pas surentraîner, j'ai préféré entraîner et évaluer mon classifieur sur des scans différents. Plus précisément, j'ai supposé que chaque scan était un partition ce qui donne une validation croisée à 3 partitions. L'entraînement se fait sur 1000 points par classe et l'évaluation se fait sur 100,000 points choisis aléatoirement. Le critère que j'optimise est le F1-score moyen. Après le processus d'entraînement, je trouve une profondeur optimale de 4, un chiffre bien plus faible que 30, nombre qu'ils trouvent dans l'article. Comme dans l'article, le score est maximisé pour 50 arbres et en utilisant le critère de Gini comme critère à minimiser.

Les scores ci-dessous sont obtenus sans les descripteurs des voisins significatifs mais avec les SC3D.

| | Sol | Bâtiment | Végét. | Barrières | Voitures | Panneaux | Moy. |
|---|---|---|---|---|---|---|---|
| F1 | 98.5 % | 89.8% | 83.3% | 1.3 % | 62.0 % | 39.0% | **62.3** % |

Table 1: Score F1 moyen obtenu en validation croisée sur le jeu de données des rues de Lille

**Paramètres et temps de calcul** Comme mentionné plus haut, je calcule des approximations des descripteurs sur 9 échelles différentes avec des voxels de côté 2.5 cm jusqu'à 6.4m. J'ai implémenté les algorithmes en Python et suis donc loin des performances de l'article en terme de temps de calcul. Pour information, les

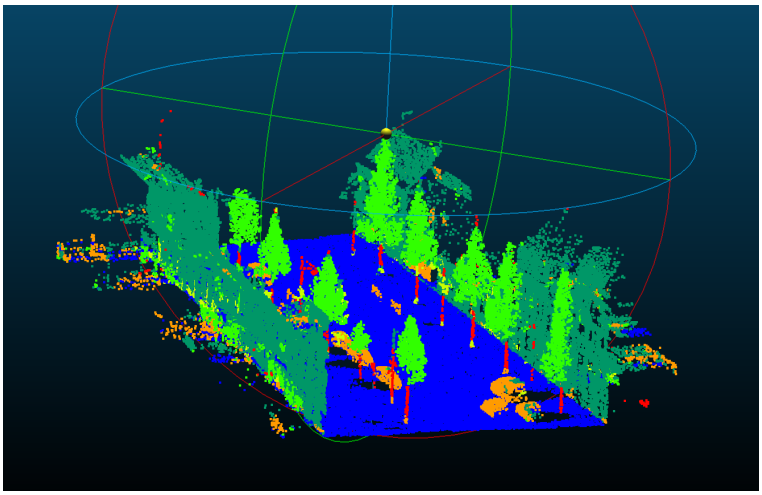

Figure 3: Prédictions pour le scan Lille3.

auteurs calculent les descripteurs sur l'ensemble du jeu de données *Paris-Rue-Cassette* (de l'ordre de $10^7$ points) en 195s. Ayant programmé en Python, pour $10^4$ points, j'obtiens des temps de calcul similaires. Les temps de calcul sont les suivants:

| | |
|---|---|
| Descripteurs basiques | 45 s |
| Descripteurs SC3D | 90 s |
| Descripteurs des voisins significatifs | 30 s |
| Entraînement du classifieur | 2 s |

Table 2: Temps de calcul des tâches principales pour 6000 points du jeu de données Lille.

# References


[1] Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bulo, and Jitendra Malik. Recognizing objects in range data using regional point descriptors. 2004.

[2] Timo Hackel, Jan D Wegner, and Konrad Schindler. Fast semantic segmentation of 3d point clouds with strongly varying density. 2016.

[3] Martin Weinmann, Boris Jutzi, Stefan Hin, and Clément Mallet. Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. 2015.