ROB317 - Vision 3D
TP3 Visualisation de nuages de points, algorithme ICP

SANTOS SANO Matheus

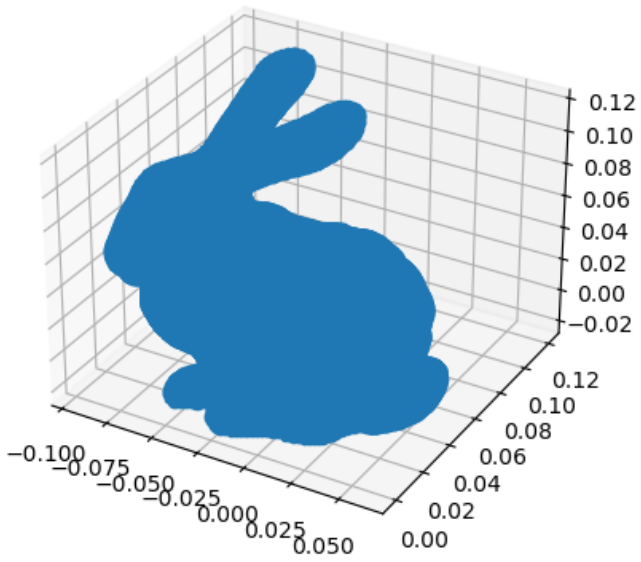
Palaiseau, France

Octobre 2023

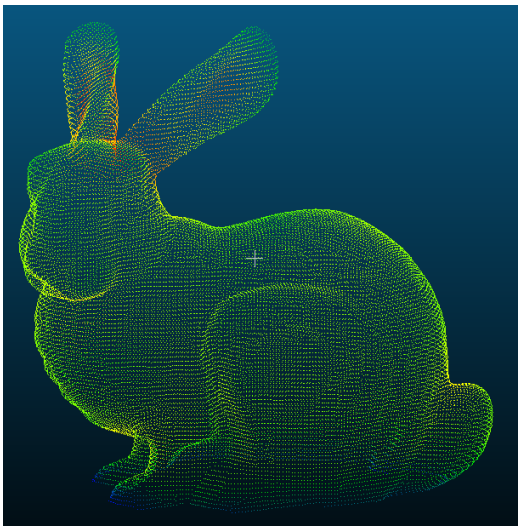
1 Visualisation de nuages de points

Ce TD vise à modifier le nuage de points par le sous-échantillonnage, la translation et la rotation. En outre, la meilleure transformation rigide entre les nuages de points est calculée et la méthode ICP est testée.

Les nuages de points manipulés au cours du TD sont présentés dans la Figure 1, le lapin de Standford et le boulevard du Montparnasse et l'église Notre Dame des Champs :

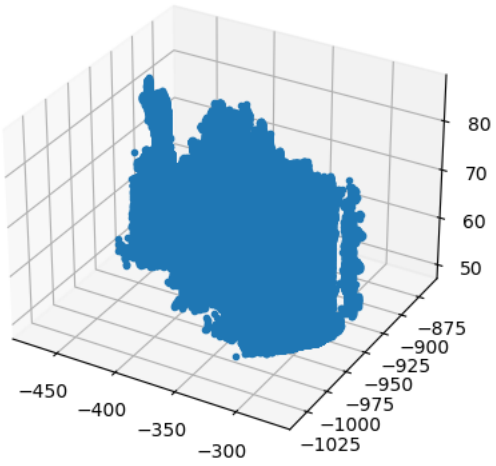


(a) Lapin de Standford lancé avec *matplotlib*.

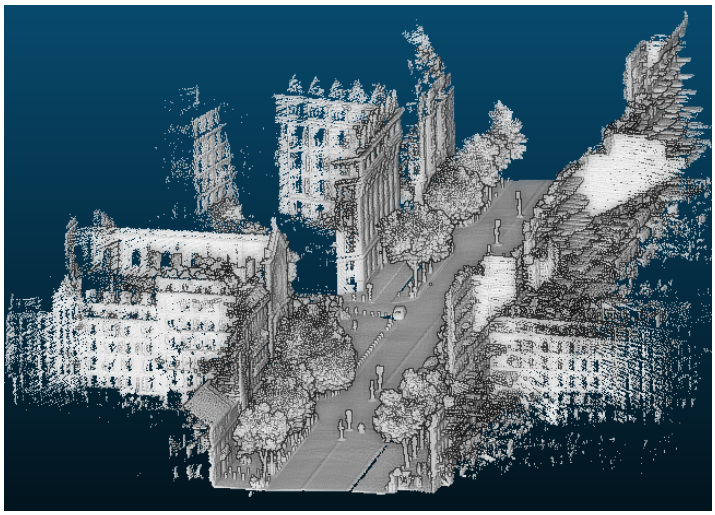


(b) Lapin de Standford lancé avec CloudCompare.

FIGURE 1 – Visualisation de nuages de points du lapin sous Python et CloudCompare.



(a) Boulevard du Montparnasse lancé avec *matplotlib*.



(b) Boulevard du Montparnasse lancé avec CloudCompare.

FIGURE 2 – Visualisation de nuages de points du boulevard du Montparnasse sous Python et CloudCompare.

Comme on peut le voir dans les figures 1b et 2b, les nuages de points présentés par CloudCompare sont plus détaillés, permettant de visualiser les détails du boulevard de Montparnasse et la forme du lapin.

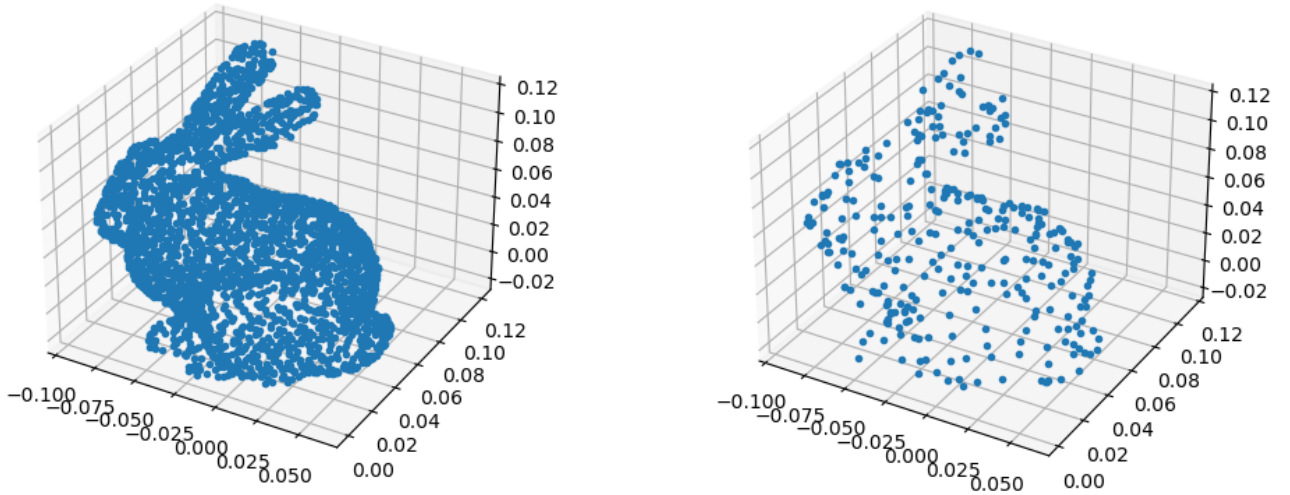
D'autre part, les nuages de points présentés par la fonction *matplotlib* de Python ne sont pas très visibles, cependant, il est possible de vérifier leurs dimensions et leur position sur l'axe des coordonnées. De cette façon, la fonction de Python sera utilisé pour analyser l'effet des transformations telles que la rotation, la translation et la transformation rigide entre nuages de points appariés.

2 Décimation de nuages de points

Soit k_{ech} le facteur de décimation et n le nombre de points, la décimation des nuages de points consiste à sélectionner n/k_{ech} points à un pas de k_{ech}. De cette manière, le sous-échantillonnage génère un nuage de points avec n/k_{ech} points. Le sous-échantillonnage des nuages de points est faite avec deux différent methodes : avec une boucle **for** et avec fonction avancée de Numpy array.

Les deux méthodes sous-échantillonnent les nuages de points, obtenant le même résultat. Cependant, comme la seconde méthode utilise les sous-tableaux de Numpy array, le temps d'exécution est beaucoup plus court que dans la méthode qui utilise la boucle **for**. La vectorisation effectuée par *Numpy* permet d'effectuer des opérations sur tout le vecteur en une seule fois, au lieu d'itérer manuellement sur les éléments comme le fait la boucle **for**. De cette manière, le temps d'exécution en utilisant la vectorisation est en fait beaucoup plus court qu'en utilisant la boucle **for**.

Alors que le temps d'exécution avec la boucle **for** est de 89.55 millisecondes, le temps d'exécution avec les sous-tableaux de Numpy array est de 0.22 millisecondes. Le résultat de la décimation pour différents facteurs d'échelle k_{ech} est :

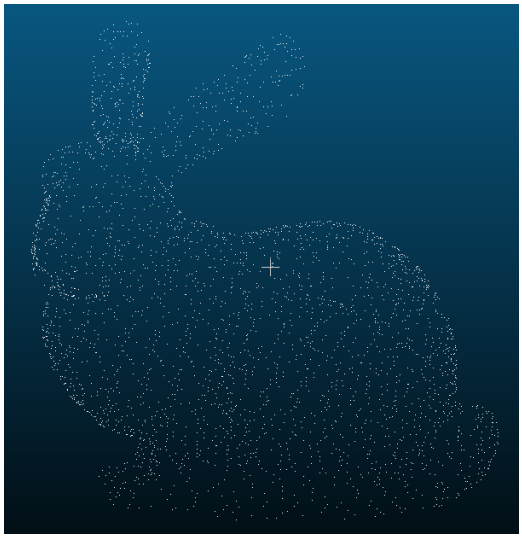


(a) Lapin de Standford après le sous-échantillonnage de $k_{ech} = 10$.

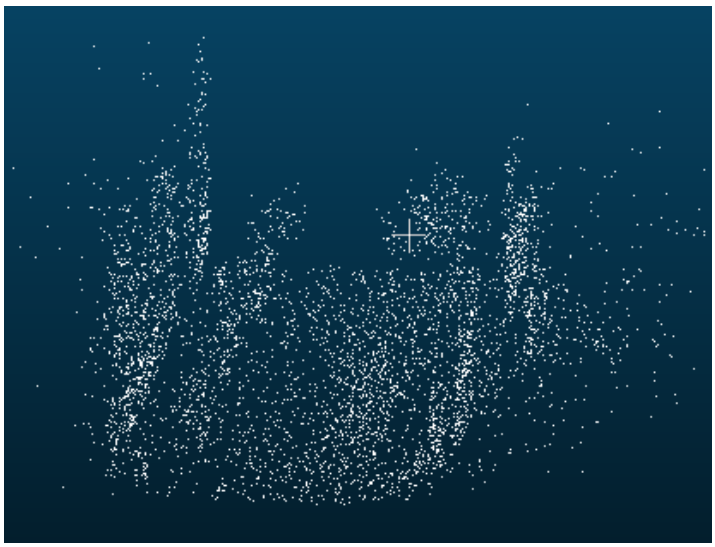
(b) Lapin de Standford après le sous-échantillonnage de $k_{ech} = 100$.

FIGURE 3 – Visualisation de nuages de points du lapin après la décimation.

Comme illustré dans la Figure 3, plus le facteur d'échelle k_{ech} est élevé, moins il y a de points sélectionnés dans le nuage de points, puisque on sélectionne un point pour chaque k_{ech} points. Le résultat de la décimation est également visible avec CloudCompare :



(a) Lapin de Standford après le sous-échantillonnage de $k_{ech} = 10$.



(b) Boulevard du Montparnasse après le sous-échantillonnage de $k_{ech} = 1000$.

FIGURE 4 – Visualisation de nuages de points après la décimation avec CloudCompare.

Comme le nombre de points du nuage de points diminue après la décimation, le lapin (Figure 4a) n'est plus aussi détaillé que dans l'image originale (Figure 1b) et le boulevard de Montparnasse (Figure 4b) n'est plus visible. Pour le rendre visible, il faut diminuer le facteur d'échelle k_{ech} et, ainsi, augmenter le nombre de points sélectionnés.

3 Translation, Rotation

Translation

La translation [0, -0.1, 0.1] effectuée sur le nuage de points original du lapin (Figure 1) est visible grâce à l'affichage sous Python. On peut voir que les coordonnées x, y et z de la Figure 5 ont, en fait, été translatées de [0, -0,1, 0,1] respectivement par rapport à la nuage de points originale (Figure 1).

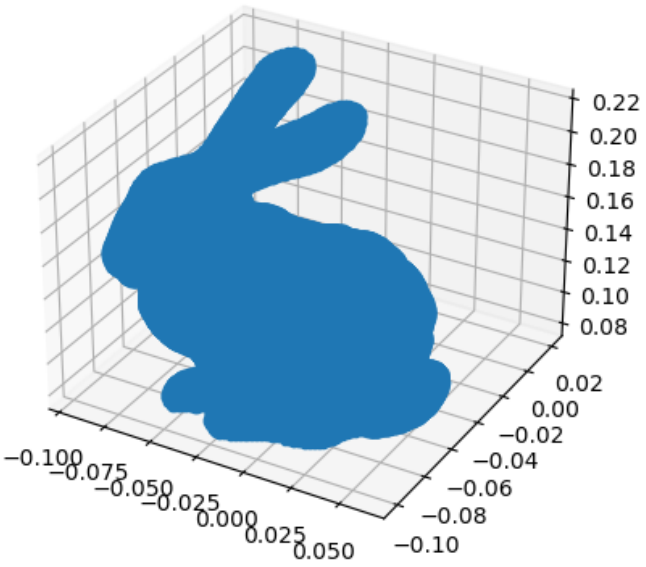


FIGURE 5 – Nuage de points du lapin translaté en [0, -0.1, 0.1].

Centrer

Pour centrer le nuage de points, la moyenne des coordonnées x, y et z est calculée et ces moyennes sont soustraites des coordonnées de chaque point. Le nuage de points du lapin centralisé est présenté à la Figure 6, où les coordonnées indiquées sont le résultat de la soustraction des coordonnées originales (Figure 5) moins la moyenne.

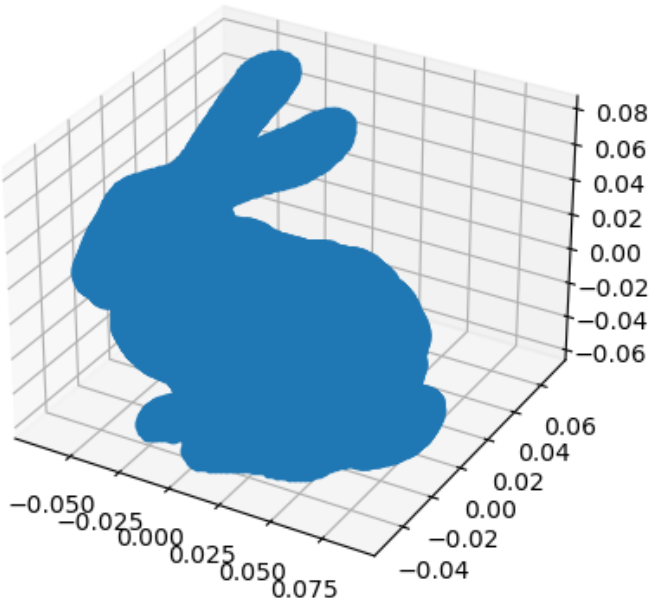


FIGURE 6 – Nuage de points du lapin centralisé.

Echelle

En divisant les coordonnées des points par une échelle de 2, le résultat est illustré à la Figure 7, où les coordonnées du nuage de points correspondent à la moitié des coordonnées originales (Figure 6).

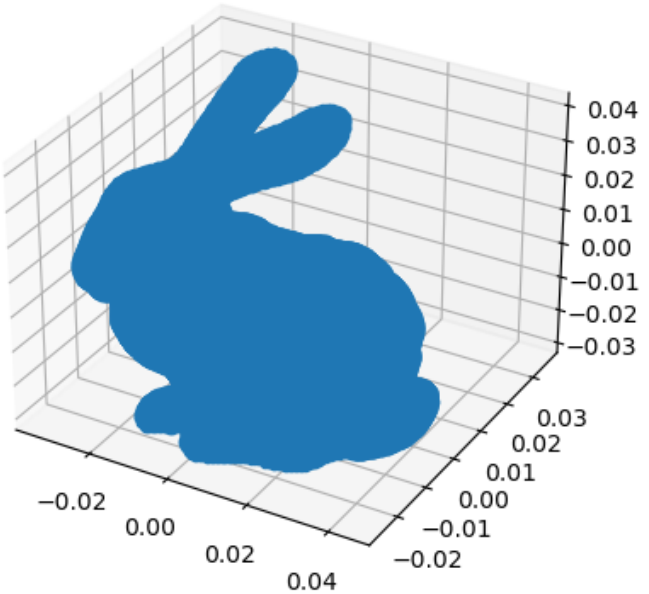


FIGURE 7 – Nuage de points après diviser les points par une échelle de 2.

Rotation

Pour appliquer la rotation sur le nuage de points centré, il faut centrer le nuage de points original (Figure 1) et, après, appliquer la rotation autour de l'axe Z avec la matrice de rotation R d'un angle $\theta = \frac{\pi}{3}$:

$$R = \begin{pmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Puis le nuage de points est re-translaté à sa position d'origine, c'est-à-dire appliquer une translation de [0, 0.1, -0.1]. Le résultat est illustré à la Figure 8, qui montre le nuage de points tourné autour de l'axe Z.

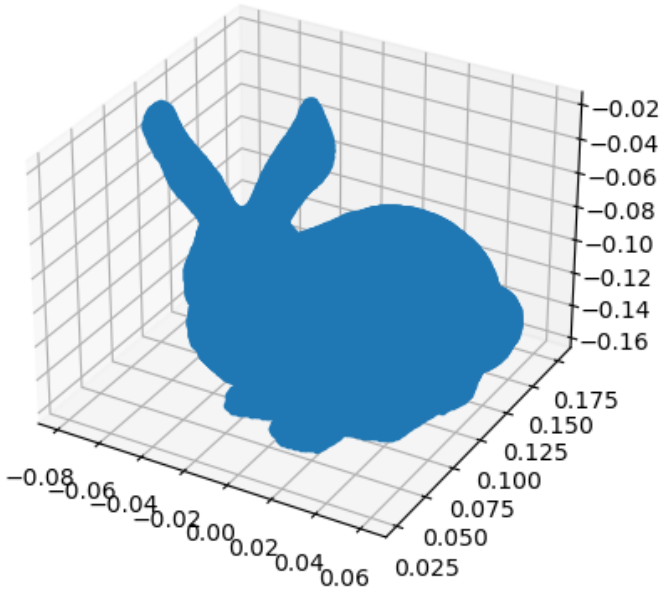


FIGURE 8 – Nuage de points tourné autour de l'axe Z.

4 Transformation rigide entre nuages de points appariés

L'algorithme du ICP (Iterative Closest Point) a pour but trouvrer la transformation rigide (R, t) entre les deux nuages de points, c'est-à-dire la rotation et la translation permettant de recaler deux nuages de points en recouvrement partiel.

À cette fin, la meilleure transformation rigide (R, t) est celle qui, par la méthode des moindres carrés, minimise la fonction $f(R, t)$:

$$f(R,t) = \frac{1}{n}\sum_{i=1}^{n}[\overrightarrow{p_i} - (R\overrightarrow{p_i}' + t)]^2, \tag{1}$$

où n est le nombre de points dans le nuage, $\overrightarrow{p_i}$ et $\overrightarrow{p_i}'$ sont les barycentres de "data" et "référence", respectivement. De cette façon, il s'agit de calculer la transformation de meilleur ajustement par la méthode des moindres carrés qui fait correspondre des points "data" à des points "référence".

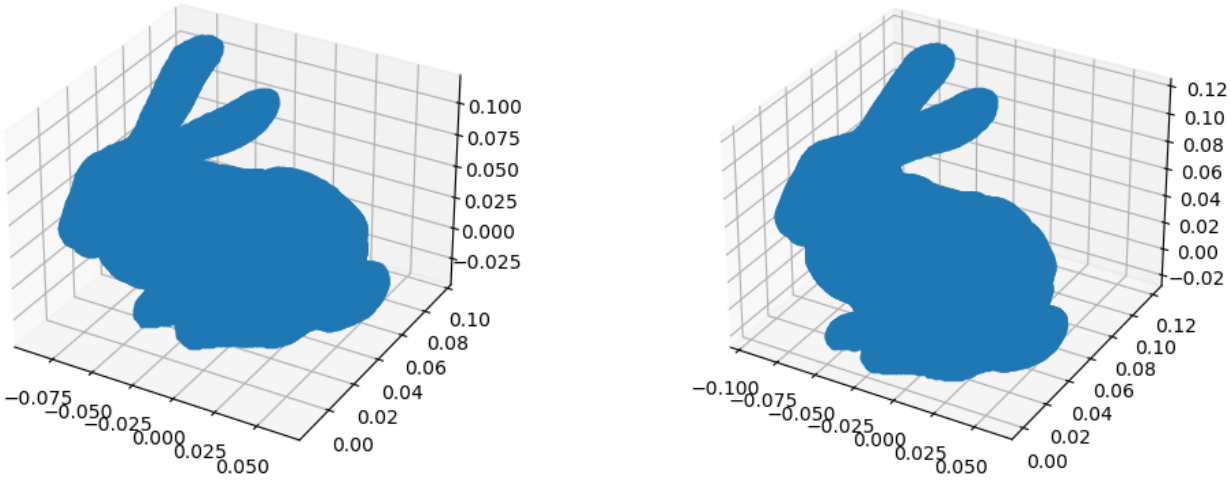
Pour ce faire, on calcule la matrice de covariance H et on fait la décomposition en valeurs singulières (SVD) de cette matrice. Soit p_m et p'_m les barycentres de les nuages de points "référence" et "data", respectivement :

$$H = \sum_{i=1}^{n} (\overrightarrow{p_i} - p_m)(\overrightarrow{p_i}' - p'_m)$$
$$[U, S, V] = svd(H)$$

La transformation rigide (R, t) est alors définie comme :

$$R = VU^T$$
$$t = p_m - Rp'_m$$

Cette transformation est ensuite appliquée au nuage de points "data", multipliée par la matrice de rotation R et additionnée à la translation t. Le nuage de points avant et après la transformation est illustré à la Figure 9.



(a) Lapin avant la transformation rigide (R, t).

(b) Lapin après la transformation rigide (R, t).

FIGURE 9 – Nuages de points avant et après la transformation rigide.

Comme le montre la figure ci-dessus, la transformation rigide transforme le nuage de points "data" (Figure 9a) en nuage de points "référence" qui est le nuage originale du lapin (Figure 1a). On peut donc conclure que la transformation rigide implémenté fonctionne correctement. Elle peut donc être appliquée à l'algorithme du ICP.

Par ailleurs, on peut vérifier si la transformation rigide trouvée est la meilleure par la méthode des moindres carrés (1). En effet, l'erreur quadratique moyenne (RMS) entre les points des nuages "data" et "référence" avant la transformation est égal à 0.019, et après, le RMS est nul, ce qui confirme que on a trouvé la meilleure transformation rigide.

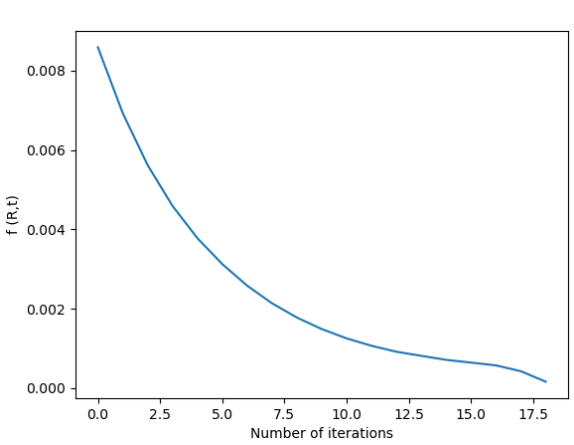
5 Recalage par ICP

L'algorithme ICP (Iterative Closest Point) trouve la transformation la mieux adaptée qui mappe les points du nuage de points "data" sur les points du nuage "référence". Cet algorithme est présenté ci-dessous :

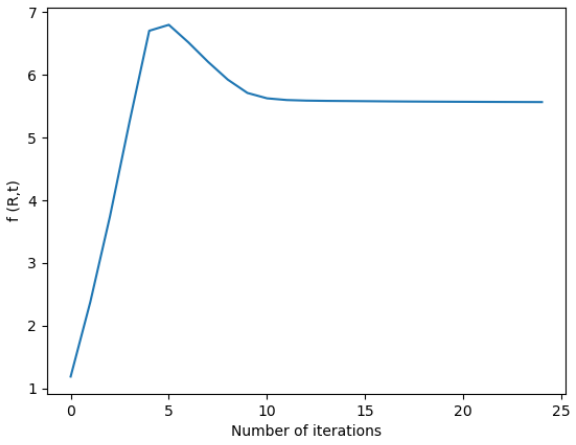
Algorithm 1: Iterative Closest Point (ICP)

Input : data, ref, max_iter, RMS_seuil
data : nuage de points à transformer avec N points de dimension d (d x N)
ref : nuage de points de référence avec N points de dimension d (d x N)
max_iter : nombre maximum d'itérations
RMS_seuil : condition d'arrêt sur RMS
Output : R, t
R : matrice de rotation
t : vecteur de translation
while *RMS*(1) *entre data et ref* $>$ *RMS_seuil* ***and*** *nombre d'itérations* $<$ *max_iter* **do**
  Détermination des points appariés (les plus proches) avec la fonction **KDTree** de *scikit-learn* ;
  Calcul de la transformation rigide (R_{temp}, t_{temp}) implementée dans la section 4 ;
  Application de la transformation au nuage "data" ;
  Calcul de la distance normalisée entre nuages RMS (1)
R = R_{temp} ;
t = t_{temp} ;

En appliquant l'algorithme ICP aux nuages de points bunny et Notre Dame des Champs (NDC 1 est "réf" et NDC 2 est "data"), la fonction $f(R,t)$ (1) est minimisée comme le montre la Figure 10 :



(a) Fonction $f(R,t)$ pour le nuage de points bunny.



(b) Fonction $f(R,t)$ pour le nuage de points Notre Dame des Champs.

FIGURE 10 – Fonction $f(R,t)$ en fonction des itérations.

Bien que cet algorithme parvienne à trouver la meilleure transformation, il est très lent si le nuage de points est très grand, comme le nuage NDC (Figure 10b). L'augmentation de la fonction $f(R,t)$, comme le montre la Figure 10b, est attribuée à la présence d'une forte densité de points dans le nuage non échantillonné. La forte densité de points proches les uns des autres entrave la capacité de la fonction *KDTree* à classer un nombre précis de points voisins pour

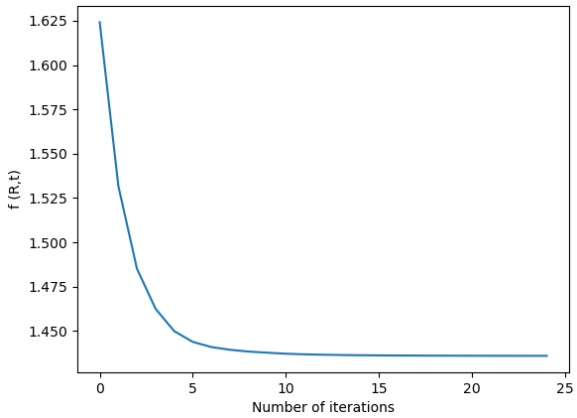
chaque point.

Une solution consiste à sous-échantillonner un des nuages ou les deux (Section 2). Ainsi, moins de points seront utilisés pour calculer la transformation (R, t) et l'algorithme ICP sera plus rapide. Par conséquent, plus le facteur de décimation est élevé, plus le temps d'exécution de l'algorithme est court. C'est ce que montre le Tableau 1.

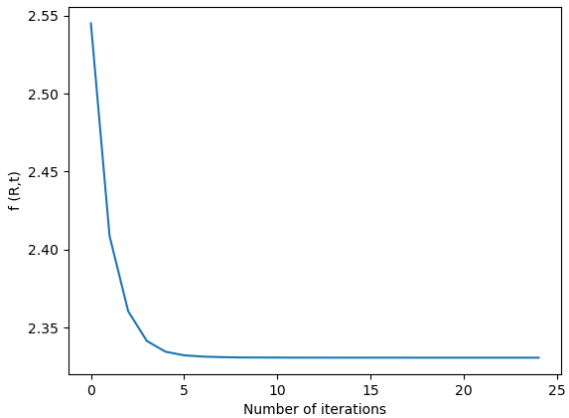
Facteur de décimation	Temps d'exécution (secondes)
sans décimation	273.97
100	0.76
1000	0.06

TABLE 1 – Temps d'exécution du ICP en fonction du facteur de décimation pour le nuage de points Notre Dame des Champs.

En plus de réduire le temps d'exécution, la sous-échantillonnage réduit également la fonction $f(R, t)$ (Figure 11a), ce qui est également souhaitable. Cependant, si le facteur de décimation est trop important, comme 1000 par exemple, le nuage de points ne conserve plus les détails du nuage original et perd sa forme. Par conséquent, il n'y a pas assez de points pour calculer avec précision la transformation rigide (R, t). Par ailleurs, la fonction **KDTree** n'est pas en mesure d'associer correctement les points entre les deux nuages de points, ce qui augmente la fonction $f(R, t)$, comme le montre la Figure 11b.



(a) Fonction $f(R, t)$ pour le nuage de points NDC avec facteur de décimation égal à 100.



(b) Fonction $f(R, t)$ pour le nuage de points NDC avec facteur de décimation égal à 1000.

FIGURE 11 – Fonction $f(R, t)$ pour le nuage de points NDC échantillonné.

De cette façon, il est nécessaire d'envisager un compromis entre le temps d'exécution et le facteur de décimation afin de trouver le meilleur facteur avec lequel il est possible d'exécuter correctement l'algorithme ICP dans un temps d'exécution pas trop long.