**Benchmarks**

1. **Plateforme du travail :**

**I. Système**: Ubuntu 20.04.2 LTS

Kernel : 5.4.0-70-generic

**II. Hardware**: Processeur : Intel(R) Core(TM) i5-1035G7 CPU @ 1.20GHz

Stockage : NVMe device Micron CT1000P1SSD8

**III. Compilateur** : g++ 9,3,0 ; ICC 2021.1 Beta 20201112

1. **Premiers Benchmarks**
2. **Exécutions indépendantes**

Dans cette section on ne collecte et enregistre que les fichiers .*ply*. On lance 200 collections consécutives puis on calcule le temps moyen utilisé

* + 1. Collection de données : *./datasaving -n 200 -i 0 -p 1 -r 0 -d 0 -f 0 3. Flags d’optimisation : -O3 -Wall -funroll-loops -march=icelake-client*.

Pour le compilateur g++ : 374,12 secondes et 1873Mb de données collectées.

Pour le compilateur ICC : 366,86 secondes et 1731Mb de données collectées.

* + 1. Lecture des nuages des points par Golang : Ici on utilise 200 fichiers en total de 1731Mb. Temps : 111,87 secondes
    2. Calculs de RANSAC itératifs : *\_, \_ = calculator.PlaneConsecRANSAC(list\_of\_vertices, 0.005, 0.1, 1000, 0.1, 1000, 5000, 500)*. Temps : 425,68s

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Collection des données | | Lecture des fichiers .*ply* | RANSAC |
| g++ | icc |
| Bande passante | 5,01 Mb / s | 4,72 Mb / s | 15,47Mb / s | 4,07 Mb / s |
| Nombre de secondes pour traiter un fichier | 1,87 s | 1,83 s | 0,56 s | 2,185 s |

Les différents compilateurs n’ont pas d’une influence significative sur la performance. La raison est peut-être que la plupart des travaux sont fait par les appels aux fonctions de la bibliothèque Realsense2, qui est déjà compilée est pré-installée.

On constate que la lecture des images est beaucoup plus rapide que l’enregistrement des données depuis la caméra. Il est probable que cette différence vient du fait que le pc utilisé est équipé d’un SSD NVME, donc la bande passante est plus élevé.

1. **Exécutions simultanées : l’enregistrement des fichiers et le traitement RANSAC**

Dans cette section, on exécute les deux processus au même temps, dont l’enregistrement et traitement de données. On lance ces processus pour une période de 200 secondes, puis à chaque intervalle de temps, on note le nombre de fichiers traités. Tous les fichiers sont de la taille et la composition similaire

|  |  |  |
| --- | --- | --- |
| Temps passé | Nombre des fichiers enregistrés | Nombre des fichiers lus et traités |
| 20s | 6 | 3 |
| 40s | 13 | 7 |
| 60s | 20 | 11 |
| 80s | 28 | 15 |
| 100s | 36 | 20 |
| 120s | 43 | 24 |
| 140s | 51 | 29 |
| 160s | 58 | 33 |
| 180s | 66 | 38 |

Dans ce cas, la bande passante pour l’enregistrement des fichiers *.ply* est de 3.17Mb/s. On constate que l’enregistrement des données est ralenti si les 2 processus sont exécutés au même temps.

1. **Profiling du programme : collection des données**
2. **Méthode de travail**

Dans cette section, on s’est servi de plusieurs outils pour réaliser un profiling du programme. Le but est de trouver une indication pour optimiser le programme afin d’obtenir une meilleure performance.

1. **Profiling**

On a utilisé le Perf et le MAQAO comme les outils principaux. Les résultats obtenus sont similaires : La fonction la plus coûteuse est *void librealsense::unpack\_yuy2<(rs2\_fo rmat)5>(unsigned char\* const\*, unsigned char const\*, int, int, int).* Elle transmet les pixels du format yuy2 dans les autres formats.

En considérant que le traitement de RANSAC et les autres algorithmes concernés ne nécessite que les informations spatiales (cordonnées en 3D). On pense à supprimer la manipulation des couleurs lors que la récupération des cordonnées des points.

1. **Amélioration du programme : collection des données**

Dans cette section, on a modifié le Realsense2 SDK de l’INTEL, en ajoutant une fonction qui exporte les fichiers de nuages des points. Le but est d’obtenir un enregistrement plus rapide pour chaque fichier.

1. **Modification réalisée**

On ajoute la fonction suivante : *void points::export\_to\_ply\_notexture(const std::basic string<char> &fname).* Cette fonction permet de collecter les nuages des points sans ajouter d’information de couleurs, qui ne sont pas nécessaire pour le traitement de RANSAC. Par conséquent, on évite la fonction coûteuse, mentionnée dans la section précédente. On prévoit une augmentation de la performance considérable car le pourcentage occupé par cette fonction est énorme.

1. **Benchmarks**

Les benchmarks sont réalisés de la même manière que la section 2.

* + 1. Exécutions indépendantes

Ici on prend 200 fichiers de nuages des points. La taille totale est de 1985 Mb. Ils sont récupérés en 109,54 secondes.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Collection des données | | |
| Avant modification | Après modification | Speedup |
| Bande passante | 5,01 Mb / s | 18,12 Mb / s | 3,617 |
| Nombre de secondes pour récupérer un fichier | 1,87 s | 0,55 s |

* + 1. Exécutions simultanées

Ici on lance le programme de collection et le programme de traitement au même temps, pendant 3 minutes. Puis on compte le nombre de fichiers traités par les deux programmes de manière ponctuelle pendant leurs exécutions.

|  |  |  |
| --- | --- | --- |
| Temps passé | Nombre des fichiers enregistrés | Nombre des fichiers lu et traités |
| 20s | 31 | 5 |
| 40s | 64 | 12 |
| 60s | 99 | 18 |
| 80s | 135 | 25 |
| 100s | 168 | 31 |
| 120s | 202 | 37 |
| 140s | 238 | 43 |
| 160s | 274 | 50 |
| 180s | 309 | 56 |

* + 1. Analyses :

On constate que l’augmentation de la performance est considérable. De plus, la perte de la vitesse en cas d’un processus de traitement lancé à côté est moins importante. Cette perte est baissée de 36.73% à 9.03%.