HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Alberto Dassatti, Bruno
Da Rocha Carvalho

Laboratoire de High Performance Coding

semestre printemps 2025

Laboratoire 4 : SIMD

Temps à disposition

6 périodes (3 séances de laboratoire)

1 Objectifs de ce laboratoire

Dans ce laboratoire, vous serez amené·e·s à analyser et à écrire du code en utilisant les instructions SIMD. Vous trouverez toutes les informations nécessaires concernant les instructions SIMD pour les processeurs Intel à l'adresse suivante : https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html

2 Analyse et amélioration

Dans cette première partie, vous devez analyser et modifier le code situé dans `part1/src/k-means.c` et `part1/include/k-means.h`, en y intégrant les optimisations SIMD que vous jugerez pertinentes.

Pour compiler et exécuter le projet, un fichier CMake est fourni. Utilisez-le de la manière suivante :

```
// Depuis le répertoire o   se trouve le CMakeLists.txt
cmake -B build         // Génère les fichiers de configuration dans le dossier "build"
cmake --build build    // Compile le projet

// Pour nettoyer les fichiers générés :
cmake --build build --target clean
```

Les sections 2.1 et 2.2 détaillent les algorithmes et leurs objectifs.

2.1 Segmentation d'image

La segmentation d'image consiste à diviser une image en segments représentant des zones distinctes. Cette technique est couramment utilisée en traitement d'image et en vision par ordinateur pour extraire des informations pertinentes.

Elle a de nombreuses applications : reconnaissance de formes, médecine, astronomie, analyse de trafic, etc. Voici un exemple d'exécution du programme fourni, segmentant une image avec 10 clusters:

Figure 1: Image originale

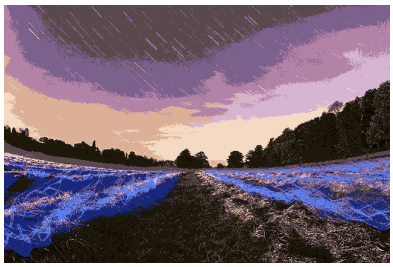

Figure 2: Image segmentée

2.2 K-Means et K-means++

K-means est un algorithme de clustering permettant de regrouper des données en k groupes distincts. Il cherche à minimiser la somme des carrés des distances entre les points et leur centre de cluster.

K-means++ est une amélioration de l'initialisation de K-means, garantissant une meilleure répartition initiale des centres, et augmentant les chances d'obtenir une solution plus proche de l'optimum global.

Dans ce laboratoire, K-means est utilisé pour segmenter les pixels d'une image selon leur teinte. Chaque pixel est ainsi associé au cluster dont la couleur RGB est la plus proche.

2.3 Code fourni

Le code proposé contient :

- Un fichier `main.c` qui charge l'image, appelle la fonction de segmentation et sauvegarde le résultat.
- Une fonction de distance :

```
float distance(uint8_t* p1, uint8_t* p2);
```

- Deux fonctions principales :

```
void kmeans_pp(struct img_t *image, int num_clusters, uint8_t *centers);
void kmeans(struct img_t *image, int num_clusters);
```

Information

Ce code est une base de travail : il permet de garantir un résultat correct et facilite les tests. Vous êtes libre de modifier, ajouter ou supprimer du code, tant que le fonctionnement général est conservé. L'utilisation de la bibliothèque `stb` pour charger les images est obligatoire.

2.4 Contraintes et conseils

- Vous pouvez utiliser d'autres images (plus grandes/complexes), mais elles doivent être au format PNG.
- Utilisation de l'exécutable : `./build/segmentation <img_src.png> <nb_clusters> <img_dest.png>`
- Vous devrez adapter le CMake pour autoriser l'utilisation d'instructions SIMD.
- Vous pouvez librement modifier les fichiers `k-means.c/.h` et `image.c/.h`.
- Les instructions SIMD utilisées ne doivent pas dépasser 256 bits (AVX2 maximum).

3 Implémentation SIMD libre d'un algorithme

Dans cette seconde partie, vous choisirez un **algorithme libre de traitement d'image** et proposerez une implémentation optimisée à l'aide des instructions SIMD. Vous devrez comparer votre solution SIMD à une version séquentielle (C standard).

Information

L'algorithme peut être simple ou complexe. L'objectif est d'expérimenter concrètement les gains que peut offrir le SIMD dans un contexte réel.

3.1 Objectifs

- Identifier un algorithme qui pourrait tirer profit d'une vectorisation.
- Implémenter une version de base (séquentielle).
- Implémenter une version SIMD à l'aide des intrinsics.
- Comparer les performances.

Avertissement

Attention aux options de compilation utilisées lors de vos tests de performance.

4 Optimisation SIMD sur votre propre code

Dans cette dernière partie, vous reprendrez votre travail réalisé lors du laboratoire sur le DTMF. Vous devez identifier **plusieurs zones pertinentes** où le SIMD pourrait apporter un gain, que ce soit au niveau des structures de données ou de l'algorithmie, et y implémenter des optimisations.

Information

Cette optimisation doit se faire uniquement sur le **décodage**, et uniquement sur **une des deux versions** développées dans le premier laboratoire.

5 Travail à rendre

Pour les trois parties du laboratoire, vous devez remettre :

- Le code modifié et compilable
- Un rapport d'analyse expliquant vos choix et les résultats obtenus

Par exemple, vous pouvez expliquer comment vous avez utilisé le SIMD pour paralléliser des opérations, ou démontrer l'amélioration de performances observée.

Information

Un bonus de **+1 point** sera attribué à l'étudiant·e ayant le code de segmentation d'image le plus performant. Les tests seront effectués sur une machine de laboratoire à l'aide de la commande `time`. Aucune instrumentation de temps n'est requise dans votre code. La performance sera évaluée en moyenne sur plusieurs images.