Parallélisation d'une nouvelle application embarquée pour la détection automatique de météores

M. Kandeepan, C. Ciocan, A. Cassagne, L. Lacassagne

LIP6, Sorbonne Université

Compas 2023

Jeudi 6 juillet 2023

Contexte

Météore : phénomène lumineux se produisant lorsque de la matière solide entre dans l'atmosphère terrestre.

Figure 1: Entrée dans l'atmosphère du vaisseau-cargo ATV (ESA).

Figure 2: Meteor Automated Light Ballon Experimental Camera (MALBEC) photo: J. Vaubaillon, IMCCE.

- Estimer le flux de matière entrant sur Terre
- Fournir des informations sur les propriétés des objets de l'Univers

Objectif : Automatiser la détection de météores à partir d'un flux video (1920×1080) en **temps réel** (25 FPS) sur un système en mouvement avec une **contrainte en puissance** ($\leq$ 10 W).

Travaux connexes

➤ FRIPON[1] :

Figure 3: Caméra FRIPON à Sorbonne Université.

- ➔ Réseau de 250 caméras réparties sur toute la France
- ➔ Caméra dirigée vers le ciel
- ➔ Aucune contrainte d'énergie
- ➔ Non robuste à la météo

[1] F. Colas et al. "FRIPON: a worldwide network to track incoming meteoroids". In: *Astronomy and Astrophysics* 644 (2020), pp. 1–23.

➤ METEORIX[2] :

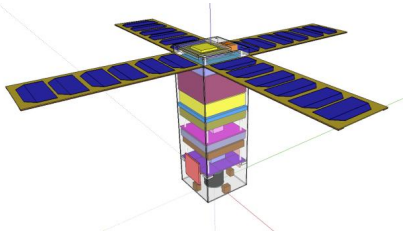


Figure 4: Nanosatellite de METEORIX.

- ➔ Projet universitaire à bord d'un **Cubesat** développé par SU
- ➔ Caméra dirigée vers la Terre
- ➔ Contraintes de faible puissance ($\leq$ 7 W) et de temps réel (25 FPS)

[2] N. Rambaux et al. "Meteorix: a cubesat mission dedicated to the detection of meteors and space debris". In: *ESA Space Safety Programme Office, NEO and Debris Detection Conference*. 2019, pp. 1–9.

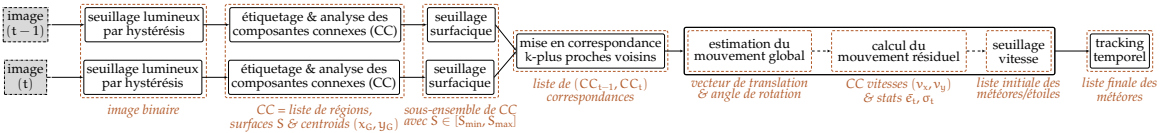
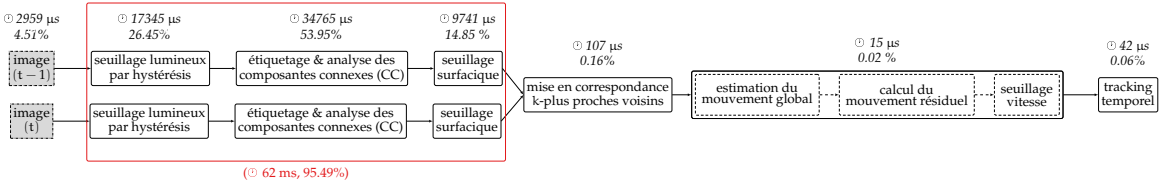


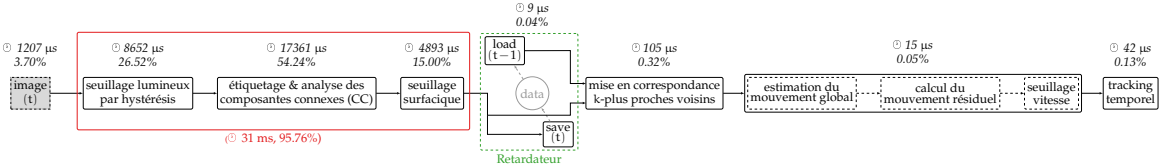
Figure 5: Chaîne de traitement de FMDT.

- Flux vidéo en niveau de gris en entrée et liste de météores en sortie
- Reconnaissance des régions d'intérêts
- Traitement robuste au mouvement du système
- Taux de détection de 85%
- Projet Open Source : https://github.com/alsoc/fmdt

Décomposition en graphe de tâches et profiling

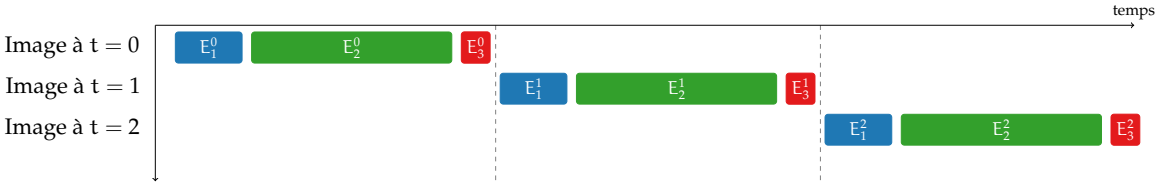


(a) Version basique.

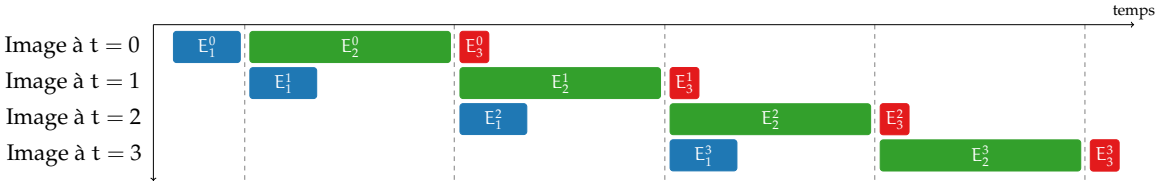


(b) Version avec mémorisation.

Figure 6: Chaîne de traitement détaillée de FMDT avec répartition du temps d'exécution des tâches en séquentiel sur un seul cœur sur Raspberry Pi 4.



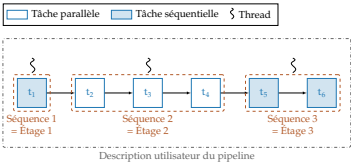
(a) Exécution séquentielle.



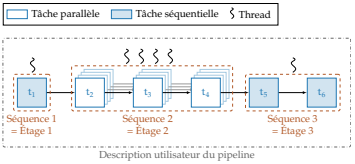
(b) Exécution à la chaîne (parallélisme *pipeline*).

Figure 7: Exécutions séquentielles et pipelinées (avec E_1^t, E_2^t et E_3^t, les étages 1, 2 et 3 à l'instant t).

Parallélisation : réplication des tâches



(a) Exécution séquentielle.



(b) Exécution à la chaîne (parallélisme *pipeline*).

Figure 8: Construction du pipeline sur plusieurs coeurs par AFF3CT[3].

- Technique qui se base le principe de pouvoir ramifier l'exécution en parallèle d'une partie du code et de pouvoir reprendre son exécution séquentielle plus tard
- AFF3CT: Langage dédié au chaîne de traitement de type streaming en C++.

[3] A. Cassagne et al. "A DSEL for High Throughput and Low Latency Software-Defined Radio on Multicore CPUs". In: *Wiley Concurrency and Computation: Practice and Experience (CCPE)* (2023). To appear. DOI: 10.1002/cpe.7820.

Parallélisation dans FMDT

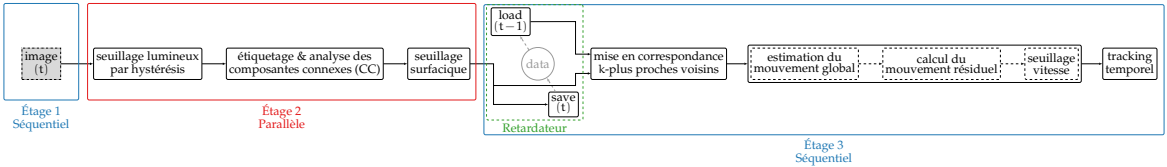


Figure 9: Chaîne de traitement de FMDT avec les étages.

Résultats : SoC évalués

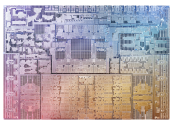
Nom complet	Date	Gravure	CPUs	Fréq.
Hardkernel Odroid-XU4	2016	28 nm	4 × *LITTLE* ARMv7 Cortex-A7 4 × *Big* ARMv7 Cortex-A15	1.4 GHz 1.5 GHz
Raspberry Pi 4 model B	2019	28 nm	4 × *Big* ARMv8 Cortex-A72	1.5 GHz
Nvidia Jetson Nano	2019	20 nm	4 × *Big* ARMv8 Cortex-A57	≈ 1.5 GHz
Apple Silicon M1 Ultra	2022	5 nm	4 × *E-core* ARMv8 (Icestorm) 16 × *P-core* ARMv8 (Firestorm)	≈ 2.0 GHz ≈ 3.0 GHz

Table 1: Spécifications des différents SoC évalués.

(a) Hardkernel Odroid-XU4

(b) Raspberry Pi 4 model B

(c) Nvidia Jetson Nano



(d) Apple Silicon M1 Ultra

Figure 10: SoC évalués.

Résultats

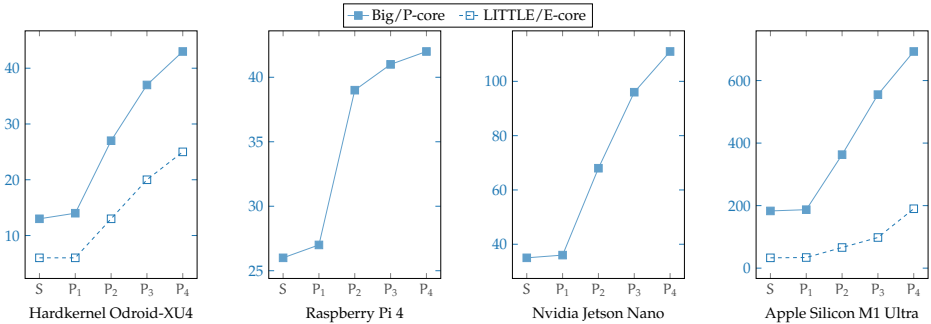


Figure 11: Débit (Images par seconde).

- ➔ S: version séquentielle avec mémorisation
- ➔ P_i: version pipeline avec i le nombre de threads de l'étage 2
- ➔ Gain insignifiant pour P_1 (*pipeline*)
- ➔ Gain significatif pour $P_{i>1}$ (réplication)
- ➔ Point critique de la Raspberry PI 4 à P_2
- ➔ **Toujours une version temps réel par SoC**

Résultats

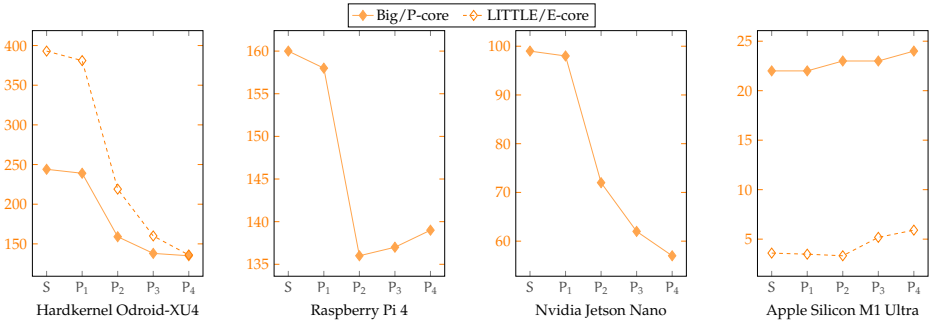


Figure 12: Énergie par image (mJ).

- ➔ La puissance est inférieure à 10 Watts pour presque tous les SoC
- ➔ En général, plus on utilise les ressources, moins on utilise d'énergie par image
- ➔ Optimum locaux pour trouver les meilleurs configurations

Conclusion

- ✓ FMDT : nouvelle application de détection de météores robuste au mouvement de la caméra
- ✓ Décomposition en graphe de tâches
- ✓ Version avec mémorisation
- ✓ Parallélisation avec pipeline et réplication de tâches
- ✓ Contrainte de temps réel respectée
- ✓ Contrainte de basse consommation respectée

Travaux futurs:

- ➤ Optimisation de certaines tâches comme le ECC/ACC[4]
- ➤ Portage des tâches régulières sur GPU

[4] F. Lemaitre, A. Hennequin, and L. Lacassagne. “How to speed Connected Component Labeling up with SIMD RLE algorithms”. In: *Workshop on Programming Models for SIMD/Vector Processing*. ACM, 2020, pp. 1–8.

Merci de votre attention !

Avez-vous des questions ?

Bibliographie

[1] F. Colas et al. “FRIPON: a worldwide network to track incoming meteoroids”. In: *Astronomy and Astrophysics* 644 (2020), pp. 1–23.

[2] N. Rambaux et al. “Meteorix: a cubesat mission dedicated to the detection of meteors and space debris”. In: *ESA Space Safety Programme Office, NEO and Debris Detection Conference*. 2019, pp. 1–9.

[3] A. Cassagne et al. “A DSEL for High Throughput and Low Latency Software-Defined Radio on Multicore CPUs”. In: *Wiley Concurrency and Computation: Practice and Experience (CCPE)* (2023). To appear. DOI: 10.1002/cpe.7820.

[4] F. Lemaitre, A. Hennequin, and L. Lacassagne. “How to speed Connected Component Labeling up with SIMD RLE algorithms”. In: *Workshop on Programming Models for SIMD/Vector Processing*. ACM, 2020, pp. 1–8.

Résultats

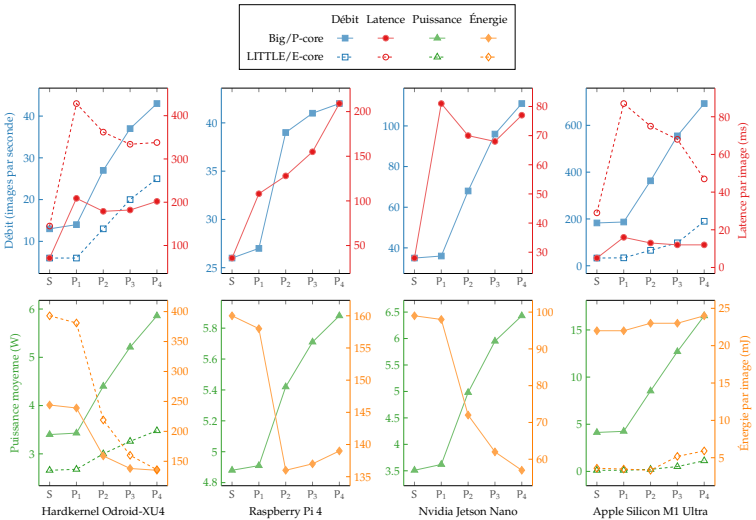


Figure 13: Performances Full HD de la chaîne de détection en terme de débit, latence, puissance moyenne et énergie consommée.

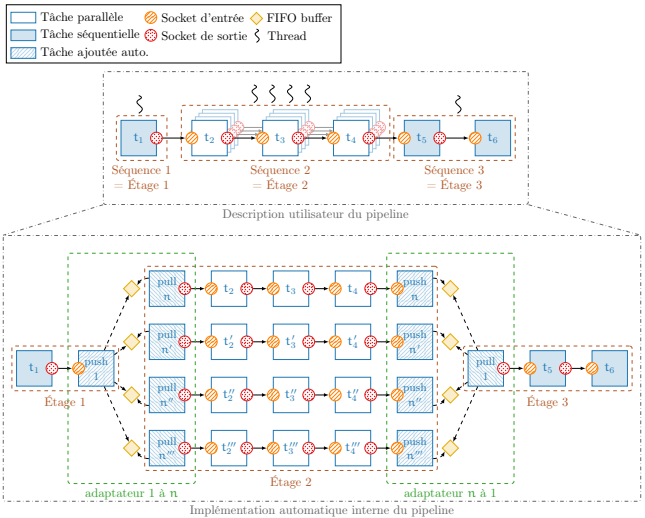


Figure 14: Construction détaillée du pipeline sur plusieurs coeurs par AFF3CT.